Development of an Image Matching Scheme Using Feature- and Area Based Matching Techniques

By Nick van der Merwe

Submitted to the University of Cape Town in fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering.

Cape Town, April 1995

Declaration

I hereby declare that this thesis is my original work and has not been submitted in any form to another university.

Nick van der Merwe
April 1995

Abstract

Image matching is widely considered to be one of the most difficult tasks of a digital photogrammetric system. Traditionally image matching has been approached from either an area based or a feature based point of view. In recent years significant progress has been made in Area Based Matching (ABM) techniques such as Multiphoto Geometrically Constrained Least Squares Matching. Also in the field of Feature Based Matching (FBM) improvements have been made in extracting and matching image features, using for example the Förstner Operator followed by feature matching. Generally, area- and feature based matching techniques have been developed independently from each other.

The aim of this research project was to design an automated image matching scheme that combines aspects of Feature Based Matching (FBM) and Area Based Matching (ABM). The reason for taking a hybrid approach is to encapsulate only the advantages of each matching scheme while cancelling out the disadvantages. The approach taken was to combine traditional aspects of ABM in digital photogrammetry with image analysis techniques found more commonly in the area of image processing and specifically machine vision.

The matching scheme utilizes a "coarse-to-fine" approach by first matching global line features, which form a "skeleton" for the subsequent matching stages. Specific point features such as corner points on the matched line features are matched next. These corresponding corner points serve as anchor points in the matching "skeleton", from where the final high-accuracy point matching can be performed. The main steps in the hybrid matching scheme are, in sequence - feature extraction, feature classification, feature matching, relative orientation and finally area based point matching. These steps are briefly summarized below:

During feature extraction the line-features representing the boundaries of objects in the image are extracted through edge detection followed by automatic line following. After edge detection the pixel coordinates of the line features are grouped using a line following algorithm and stored. This algorithm has been adapted to fill in gaps appearing in edge chains. A feature editing step has been provided, which constitutes the only semi-automatic step in the whole matching scheme. After vectorizing the pixel level feature boundaries using line following, the feature boundaries are calculated to the sub-pixel level.

Feature classification assigns attributes to a feature which enables the image analysis algorithm to uniquely describe each feature and use this description during the feature matching stage. A feature descriptor called the $d\phi(s)$ plot was used as feature "signature" function. From this descriptor function other attributes such as the size, type and number of corners of the line feature can be deduced. A new "circle indicator" is described, which indicates circular features using a statistical analysis of the signature function. A new feature descriptor function, the "Gradient-s" plot is also presented.

Feature matching depends on the results of the feature classifications stage. Candidates for feature matching are obtained by comparing the attributes of features. Matching candidates are matched using a novel two-stage matching scheme. Firstly, the signature functions of the feature pairs are correlated through a normalized cross-correlation function using FFT's. The second step uses the results

of this correlation to find the correct match by examining the feature topology using matching triangles. Triangles are formed between the centre-of-mass points of the reference feature and the two nearest features. Possible feature matches are verified by searching for similar triangles formed in the candidate feature-space.

The relative orientation is calculated automatically from the matched features. The first rough relative orientation is calculated using the centre-of-mass points of matched line-features. Using the resultant epipolar condition and matching triangles more features are matched, and the corresponding centre-of-mass points are used to update the relative orientation. The final relative orientation is obtained by matching corresponding corner points of the matched features to a high degree of accuracy using Least Squares Matching and using this information to update the relative orientation.

Finally, the high-accuracy point matching can be performed using the matched line-features and corner points as reference framework. Points on a feature outline can be matched using Geometrically Constrained Least Squares Matching. The initial approximation to a matching point is obtained by searching for the intersection between the epipolar line and the corresponding feature boundary.

The results obtained proved that the matching scheme worked well from a systems-development point of view, and it is envisaged that with further research this matching scheme can be applied for more general applications.

Acknowledgements

I am extremely grateful to my supervisor and mentor, Prof. Heinz Rüther, for his dedication and guidance throughout this project. Not only his sound advice and blunt criticism was highly appreciated, but also the fact that I was given more than ample room to investigate my own ideas and approach. For creating time for discussion, teaching, argument and specifically for reviewing this thesis dissertation I am more than grateful to him.

I am very thankful to have had the guiding hand of my colleague and friend Graham van der Vlugt throughout my postgraduate studies. From Graham I have learned an immense deal, and I am extremely grateful to him for always having time to give sound advice.

I would also like to thank my other colleagues in the Department of Surveying and Geodetic Engineering at UCT for their support. Specifically, I would like to mention Julian Smith, Michael Calitz, Sue Binedell, the secretaries Val and Lee and administrative staff Michael and Sidney.

To my friends and colleagues in the Department of Electrical Engineering at UCT I would like to extend a big thank you. Specifically to Anthony Kanfer and Greg Cox for their constructive criticism, ideas and proofreading of my often garbled "memoirs." I would also like to extend my appreciation to Fred Hoare for allowing me access to the Suns in the digital image processing laboratory and to my system administrator, Tony Donno, for his help in solving many problems.

Other people in this department whom I would like to thank are Prof. Gerhard de Jager for his advice and review of the areas that are based in digital signal- and image processing. Thanks also to Brendt Wohlberg, Peter Moon and Robert Crida for their advice.

I would not have been able to complete this thesis without the love and support of my family, and I am extremely grateful to them. I would like to thank my mother for her love, interest and at times desperately needed financial support. I know my late father would have been proud of me, and I dedicate this thesis to his memory.

Finally I would like to thank my girlfriend Alison for her love and support over the last five years. Even during the last year during which circumstances forced us apart her love and long range support served as a major inspiration.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background - The Image Matching Problem

The approach to photogrammetry and image interpretation has been changed dramatically due to vast strides in computer- and image acquisition technology. Improvements in imaging sensors, scanning devices, computer storage, computer architecture and processing power have, collectively, increased the accuracy, reliability and scope of digital photogrammetric systems. Of vital importance, however, is the capability of software algorithms to also improve in order to take full advantage of the emerging technologies. The combination of hardware and software has lead to the development of precise and reliable techniques for non-contact 3-D measurements, with applications in science, art and industry.

In the keynote paper at the ISPRS Commission V Conference in 1994, Gruen [39] reflects on the progress of specifically digital close-range photogrammetry over the last ten years. He sees a digital close-range photogrammetric system as a "multi-eyed measurement robot", whose most essential component is a vision system used to analyze the environment. Gruen specifies that a "photogrammetric" system should meet the following requirements:

- Self-diagnosis capability
- Potential for high precision and reliability
- Task flexibility with respect to 3-D object reconstruction functions

The task of the photogrammetric system is to convert the unstructured, raster image of a scene into a meaningful and structured vector representation. This vector representation can be used for the interpretation of geometric- and spectral properties and semantic information about objects in the real world.

Heipke [50] concentrates on the transformation of the analytical stereo processing system to a digital stereo processing system. A distinction can be made between completely automatic processing tasks and semi-automatic tasks. Heipke cites the examples of image preprocessing,

interior and relative orientation and point transfer among others as being fully automated using batch processes. In some cases not all tasks of a digital photogrammetric system can be automated, and user input is needed to supervise the semi-automatic identification and measurement process. According to Gruen [39] full automation is only required in on-line applications, which usually make use of artificial structure such as retro-targets or structured light in a well controlled environment. In satellite and aerial scenes the image analysis algorithms often have to deal with "natural" features such as edges, corners and other composite structures, which poses great demands on the system. In these cases user input is needed to guide the semi-automatic processing of the input data.

Barnard and Fischler [8] defined six steps necessary for stereo analysis:

- Image Acquisition
- Camera Modelling
- Feature Acquisition
- Image Matching
- Depth Reconstruction
- Interpolation

Of these steps, *image matching* is widely considered to be the most difficult to solve. Hahn [44] describes image matching as "..the task of finding the correspondence between a description of some images of a scene or of finding the correspondence between a description of the images and a description of the model (including projection) of the scene". Modelling, search and localization are considered by Hahn to be the main aspects of the image matching procedure.

The two main research directions in image matching are Area Based Matching (ABM) and Feature Based Matching (FBM). Cochran and Medioni [17] make the following distinction between the two matching schemes: "When the matched features are low level and dense, such as the intensity at each pixel, and the matching strategy is applied locally throughout the image at each pixel, we call the matching strategy *area based.* When a few interesting (usually more abstract) features are first selected from the image and the matching strategy is applied to this subset, we use the term *feature based.*" In area based matching the image intensities are matched directly using techniques such as the normalized cross-correlation or Least Squares Matching (LSM). In feature based matching prominent point or line features such as corners or edges are extracted first and then matched in a subsequent step using a "correspondence generator", which tries to minimize the differences between the two feature representations using various optimization techniques.

Area based matching techniques have successfully been implemented in the analysis of aerial terrain images, where the surface typically varies smoothly. An advantage of ABM is that it directly produces a dense disparity map and thus results in a dense point field which can be used to build for example a Digital Terrain Model (DTM). A disadvantage of this matching scheme is that it usually breaks down when there is a lack of texture or where depth

discontinuities occur. Feature based matching is more suited to the analysis of scenes that have some structure present. In for example aerial scenes of cultural (man-made) features or close-range scenes of industrial parts there are generally many features such as corners and lines that can be extracted. More complex features such as unique combinations of straight lines, circles, rectangles and general polygons can often also be extracted and used in the feature based matching scheme. Using image features as matching primitives has the advantage of decreasing the search-space for possible matches, but typically results in a sparse point field of corresponding points.

1.2 Problem Statement for this Research Project

The aim of this research project was to design an automated image matching scheme that combines aspects of Feature Based Matching (FBM) and Area Based Matching (ABM). The reason for taking a hybrid approach is to encapsulate only the advantages of each matching scheme while cancelling out the disadvantages of each scheme. Using such a hybrid approach has been proposed by researchers such as Hahn [44] and Cochran and Medioni [17] amongst others.

Specific features of interest have to be extracted and analyzed in each image of a stereo-pair of images from close-range or aerial images. Using the results of the image feature analysis the features have to be matched and these corresponding features used as a framework for finding corresponding image points.

The complete matching system has to be automated as far as possible, preferably with no human input at all.

The following assumptions and criteria were specified:

- The interior orientation parameters for each image in the stereo-pair are known.
- No external control points and thus exterior orientation information is available.
- The input images are rich in structure and have a reasonably dense spread of image features.
- The image analysis step should be able to form a distinction between the "background" and relevant "foreground" structures in an image.
- The image matching scheme should preferably be PC-based, with speed a non-vital criterion.

The matching scheme has to be tested using suitable example images and the success of specifically the feature matching stage evaluated. The three-dimensional coordinates of specific image point features have to be calculated and the discrete measurements between such point features obtained.

The research should make a relevant contribution to the area of image matching, and specifically to the development of hybrid approaches. Specific areas suitable for further research should also be specified.

1.3 Organization of this Dissertation

The thesis starts with a general overview of the fundamental concepts used in digital photogrammetry and in image processing.

Chapter two explains the fundamental concepts used in digital photogrammetry. The collinearity equations are derived and models for calculating the interior, relative and absolute orientation between two images are discussed.

Chapter three briefly introduces some of the basic concepts used in digital signal- and image processing. Reference is made to the use of standard image processing techniques in digital photogrammetry.

In *chapter four* the extraction of features used in image matching is described. Some standard edge detection techniques are presented and the extraction of line features using line-following is discussed. A novel method of improving the edge detection results using "edge strengthening" is presented and a unique adaption of an existing line-following algorithm to fill in gaps is described.

Chapters five and six together form the main part of this dissertation. In these chapters the most important new contributions of the image classification and the unique feature matching algorithm are presented.

Chapter five deals with the classification of features. Extracted features are used for image matching, and the classification of features is a vital step in the feature based matching process. The classification of line features using existing and newly developed one-dimensional descriptor functions is discussed, followed by a brief review of other existing feature descriptors. In this chapter a new "circle indicator" is described, which indicates circular features using a statistical analysis of the signature function.

Chapter six deals with the matching of features, and thus is the most important chapter in this dissertation. In this chapter the selection of possible feature matching candidates using the results of the feature classification is discussed. The selected candidates are then matched using a newly developed two-step matching process. The first step in this matching process, the correlation of the feature descriptor functions, is described and analyzed. Next the second step of the feature matching process using matching triangles to check feature topology is presented. A novel method for evaluating the correspondence between two matching triangles is presented.

In *chapter seven* the automatic calculation of the relative orientation between images is discussed. The relative orientation calculation is based on corresponding pairs of image points, and in this chapter the solution to the relative orientation using corresponding pairs of image points is described. The calculation of the epipolar lines for image points will also be dis-

cussed. Using the centre-of-mass points of matching features as corresponding points forms a new contribution to the area of automatic image registration.

In *chapter eight* area based matching techniques will be discussed. Least Squares image matching will be described first, followed by an explanation of the incorporation of the epipolar condition to geometrically constrain the image matching process. A novel approach to simplify the final area based matching process by combining the matched features with the epipolar condition is presented. The matched features combined with the epipolar condition are used to find the starting positions for image patches used in area based matching, and specific reference will be made to the matching of feature corner points using this method.

In the final chapter some conclusions will be drawn based on this research and recommendations made for aspects of further study.

Chapter 2

Fundamentals of Digital Photogrammetry

Photogrammetry is defined by the American Society for Photogrammetry and Remote Sensing (ASPRS) as "*The art, science and technology of obtaining reliable information about physical objects and the environment by recording, measuring and interpreting photographic images*" [31]. Photogrammetry thus entails a) capturing an image of the object, b) measuring the processed image of the object and c) reducing the measurements and spatial information to some useful form such as a topographic map or a Digital Terrain Model (DTM) [77].

Two broad classes of photo interpretation can be identified :

1. *Quantitative*, which involves precision dimensional measurements such as space coordinates, ground positions, elevations, distances and volumes. This aspect of photogrammetry requires the setting up and solution of suitable mathematical models.
2. *Qualitative*, which deals with the recognition and interpretation of objects. Digital, graphic or visual representation is usually implied.

Photogrammetry has progressed from the traditional approach of interpreting aerial photographs to the interpretation of digital images. Vast improvements in computer technology have changed the face of photogrammetry from the analog world of aerial photographs and topographic maps to digital images and DTM's. Imaging sensors such as Charge Coupled Device (CCD) cameras are capable of directly producing an image in a digital format that can be manipulated using computers. The advent of high quality scanners has also made it possible to convert an analog photograph into a high-resolution digital format. Digital images can be processed by powerful computers with a minimum of human interaction. This process is termed *Digital Photogrammetry* and is defined by Haralick and Shapiro as "*The computer processing of perspective projection digital images with analytic photogrammetry techniques as well as other computer techniques for the automatic interpretation of scene or image content*" [47].

In this chapter the mathematical models for the image geometry and the transformations

from the image to the object are briefly described. The collinearity equation, which forms the basis of the mathematical model, is derived. The concepts of *interior-*, *relative-* and *absolute* orientation are introduced and the parameters of the relevant mathematical models are defined.

2.1 Image Geometry

The mathematical model that relates the object in three-dimensional space (*object space*) to the projection of that object on the image (*image space*) is referred to as the *Perspective Projection*. Refer to figure 2.1 for the perspective projection of the letter **F** from object space to image space.

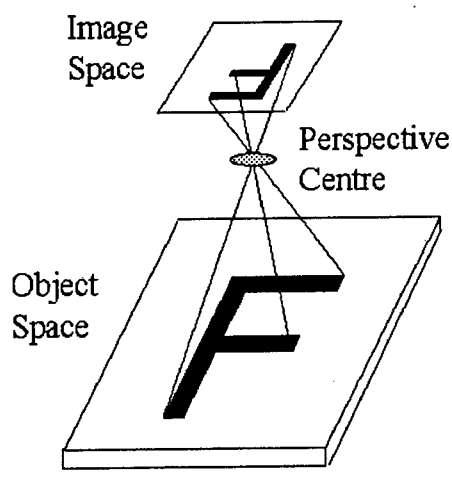


Figure 2.1: Perspective Projection of the letter F

The theoretical thin lens has a centre through which all incident light rays pass without deviation. According to Scott [91], the *perspective centre* of the central projection must be divided into an object space point and an image space point, called the outer and inner perspective centres. Scott proves in his work that these points do not necessarily coincide with the front and rear nodes of the camera lens. The image is formed on the image plane, which is a distance f from the inner perspective centre. The constant f is referred to as the *focal length.*

The geometric configuration for two cameras at different positions observing the same object point in three-dimensional space is shown in figure 2.2. This is the basic geometric setup used in analytical photogrammetry.

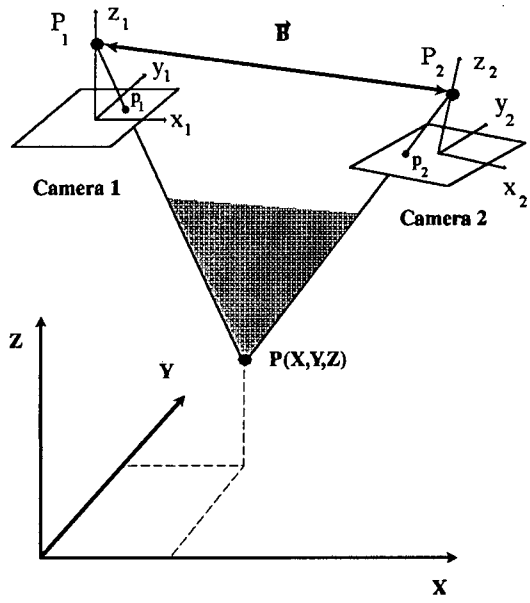


Figure 2.2: Geometric Configuration for a 2-Camera Imaging System

The coordinate system used in object space is a right-handed coordinate system in X, Y and Z. The object point $P(X,Y,Z)$ is projected onto the positive image plane of camera 1 at the point in image space labelled as p_1. The same object point $P(X,Y,Z)$ is projected onto the positive image plane of camera 2 at the point indicated as p_2 in image space. Finding the transformation from the object point $P(X,Y,Z)$ to the image points p_1 and p_2 allows us to translate measurements from image space to object space, which is the basis of analytical photogrammetry.

The perspective centres for the two cameras are indicated as P_1 and P_2, with object space coordinates (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) respectively. The image space coordinate system x_1, y_1 and z_1 for camera 1 has its origin at the perspective centre P_1 and the image space coordinate system x_2, y_2 and z_2 has origin at P_2. The principal point of the image is defined in the ASPRS Manual of Photogrammetry [105] as *the foot of the perpendicular from the interior perspective centre to the plane of the photograph.* This point is the projection of the perspective centre on the image plane, and image coordinates are measured relative to this point. The principal distance is defined as *the perpendicular distance from the internal perspective centre to the image plane.* Often the term "focal length" is used synonymously with principal distance, but strictly speaking the focal length refers to the principal distance at infinity focus. In the case of close range photogrammetry however, the image is not obtained at infinity focus. The calibrated focal length is the value of the focal length that produces an overall mean distribution of radial lens distortion [28].

For any camera with perspective centre (X_0, Y_0, Z_0), principal point (x_p, y_p) and principal distance f the transformation from image space coordinates x, y and z to object space coordinates X, Y and Z can be modelled by three translations, three rotations and a scale i.e.

$$\begin{pmatrix} x - x_p \\ y - y_p \\ -f \end{pmatrix} = s\, R_{(\omega,\phi,\kappa)} \begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix} \tag{2.1}$$

where (x_p, y_p) is the projection of the perspective centre on the image plane.

Image coordinates are reduced to the principal point with the corresponding object space coordinates reduced to the camera perspective centre. The scale s relates the relative distances between image space and object space. The rotation matrix R depends on the rotations of the image plane about the object space axes. Figure 2.3 shows the right handed object space coordinate system with the camera rotations shown.

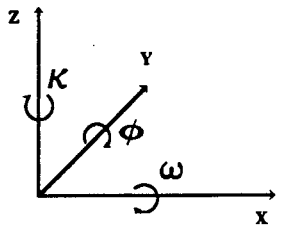


Figure 2.3: Object Space coordinate system with rotations ω, ϕ and κ

The rotation angle ω is measured around the X-axis, ϕ is measured around the Y-axis and κ around the Z-axis. The transformation matrices for each of these individual rotations are defined as follows:

$$R_X(\omega) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\omega & sin\omega \\ 0 & -sin\omega & cos\omega \end{pmatrix}$$

$$R_Y(\phi) = \begin{pmatrix} cos\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{pmatrix}$$

$$R_Z(\kappa) = \begin{pmatrix} cos\kappa & sin\kappa & 0 \\ -sin\kappa & cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

To obtain the overall rotation matrix the effect of the three rotation matrices R_X, R_Y and R_Z is combined by multiplying the three rotation matrices together i.e. $R_{(\omega,\phi,\kappa)} = R_Z(\kappa)R_Y(\phi)R_X(\omega)$. The assumption made here is that the rotation ω about the X-axis is taken into account first, then ϕ and then κ. The order in which the matrices are multiplied affects the outcome of the resultant rotation matrix R.

$$R_{(\omega,\phi,\kappa)} = \begin{pmatrix} cos\phi\, cos\kappa & sin\omega\, sin\phi\, cos\kappa + cos\omega\, sin\kappa & -cos\omega\, sin\phi\, cos\kappa + sin\omega\, sin\kappa \\ -cos\phi\, sin\kappa & -sin\omega\, sin\phi\, sin\kappa + cos\omega\, cos\kappa & cos\omega\, sin\phi\, sin\kappa + sin\omega\, cos\kappa \\ sin\phi & -sin\omega\, cos\phi & cos\omega\, cos\phi \end{pmatrix} \tag{2.2}$$

For simplification this matrix is assigned symbolically:

$$R_{(\omega,\phi,\kappa)} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

2.2 The Collinearity Equations

The *collinearity equations* are the most fundamental equations in analytical photogrammetry, and are used as the basis for most of the calculations done in photo interpretation.

By dividing through with the Z component of equation 2.1 the collinearity equations are obtained

$$\frac{x - x_p}{-f} = \frac{r_{11}(X - X_0) + r_{12}(Y - Y_0) + r_{13}(Z - Z_0)}{r_{31}(X - X_0) + r_{32}(Y - Y_0) + r_{33}(Z - Z_0)} \tag{2.3}$$

$$\frac{y - y_p}{-f} = \frac{r_{21}(X - X_0) + r_{22}(Y - Y_0) + r_{23}(Z - Z_0)}{r_{31}(X - X_0) + r_{32}(Y - Y_0) + r_{33}(Z - Z_0)} \tag{2.4}$$

Due to the division with the Z component the scale s has been cancelled out.

These equations are commonly simplified as

$$(x - x_p) = -f\frac{M_1}{M_3}$$

$$(y - y_p) = -f\frac{M_2}{M_3}$$

where
$M_1 = r_{11}(X - X_0) + r_{12}(Y - Y_0) + r_{13}(Z - Z_0)$
$M_2 = r_{21}(X - X_0) + r_{22}(Y - Y_0) + r_{23}(Z - Z_0)$
$M_3 = r_{31}(X - X_0) + r_{32}(Y - Y_0) + r_{33}(Z - Z_0)$

The rotation matrix $R_{(\omega,\phi,\kappa)}$ is an orthogonal matrix and thus has the specific property that the inverse R^{-1} is the same as the transpose R^T. To transform from image space back into object space the image coordinates in equation 2.1 are multiplied by the inverse rotation matrix R^T.

$$\frac{X - X_0}{Z - Z_0} = \frac{r_{11}(x - x_p) + r_{21}(y - y_p) - r_{31}f}{r_{13}(x - x_p) + r_{23}(y - y_p) - r_{33}f} \tag{2.5}$$

$$\frac{Y - Y_0}{Z - Z_0} = \frac{r_{12}(x - x_p) + r_{22}(y - y_p) - r_{32}f}{r_{13}(x - x_p) + r_{23}(y - y_p) - r_{33}f} \tag{2.6}$$

In digital photogrammetry the mathematical model that the collinearity equations are based on is extended to also take into account *additional parameters.* Beyer [11] groups additional parameters as "parameters modelling the interior orientation, parameters to model inadequacies of solid-state sensors and synchronization, parameters modelling lens distortion, parameters to model other effects such as deviations from a flat surface, and parameters to absorb arbitrary deformations". In this thesis only parameters modelling lens distortion are discussed. Lens distortion is discussed in more detail in section 2.3.1 on interior orientation.

2.3 Camera Calibration and Orientation

To accurately interpret distances on images during photo-interpretation, the interior and exterior geometry of the sensing device such as an aerial- or CCD camera must be known to a high degree of accuracy. The camera geometry for a stereo-pair of images can be expressed in terms of the *interior*, *relative* and *absolute* orientations.

2.3.1 Interior Orientation

The interior orientation of a camera refers to the internal geometry of a bundle of rays incident on the image plane. The perspective geometry of the camera can be modelled by the following parameters:

1. The calibrated focal length - defined as the principal distance f
2. The position of the principal point (x_p, y_p) in the image
3. The geometric distortion characteristics of the lens system

The principal point and principal distance were defined earlier in section 2.1.

Refer to figure 2.4 for the interior orientation geometry. The image coordinates are reduced to the principal point (x_p, y_p) on the image plane s.t.

$$x' = x - x_p$$

$$y' = y - y_p$$

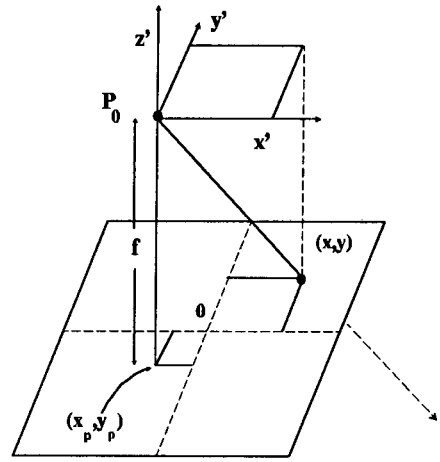


Figure 2.4: Image Geometry for Interior Orientation

In conventional photogrammetry the centre of the image at 0 is defined by the intersection of the lines linking the fiducial marks. In digital photogrammetry this origin point is defined as the centre of the image if no fiducial marks are available.

Of the additional parameters mentioned only the *lens distortion* parameters were taken into account in this thesis project. The most commonly used lens distortion characteristics are those modelled by Brown using three radial and three tangential distortion parameters [29] [11] [105].

The corrections dx and dy to the image coordinates can be expressed as

$$\begin{aligned} dx &= \{k_1 r^2 x' + k_2 r^4 x' + k_3 r^6 x'\} + \\ &\quad \{p_1[r^2 + 2x'^2] + 2p_2 x'y'\} \end{aligned} \tag{2.7}$$

$$\begin{aligned} dy &= \{k_1 r^2 y' + k_2 r^4 y' + k_3 r^6 y'\} + \\ &\quad \{p_2[r^2 + 2y'^2] + 2p_1 x'y'\} \end{aligned} \tag{2.8}$$

where k_1, k_2 and k_3 are the radial distortion parameters and p_1 and p_2 are the decentering distortion parameters.

The radius r in equations 2.7 and 2.8 is defined as the distance between the image point (x, y) and the principal point (x_p, y_p) i.e.

$$r = \sqrt{(x - x_p)^2 + (y - y_p)^2} \tag{2.9}$$

The image coordinates are corrected for lens distortion dx and dy by

$$\bar{x} = x' + dx \tag{2.10}$$

$$\bar{y} = y' + dy \tag{2.11}$$

In digital photogrammetry image coordinates are readily available in pixel coordinates. To apply the mathematical models derived these pixel coordinates have to be transformed to distances in metric units such as millimetres.

To transform from pixel coordinates (x, y) to image coordinates (u, v) in millimetres the following transformations are applied:

$$u = ps_x(x - x_c) - x_p \tag{2.12}$$

$$v = ps_y(y_c - y) - y_p \tag{2.13}$$

where:

ps_x	=	pixel spacing in the x-direction with units mm/pixel
ps_y	=	pixel spacing in the y-direction with units mm/pixel
(x_c, y_c)	=	the image centre in pixel coordinates
(x_p, y_p)	=	the principal point coordinates in mm

For a typical 512x512 CCD image captured with a PIP 512-B framegrabber the centre of the image in pixels is (255.5,255.5) with pixel spacings in x and y of approximately 10-15 micron per pixel [11].

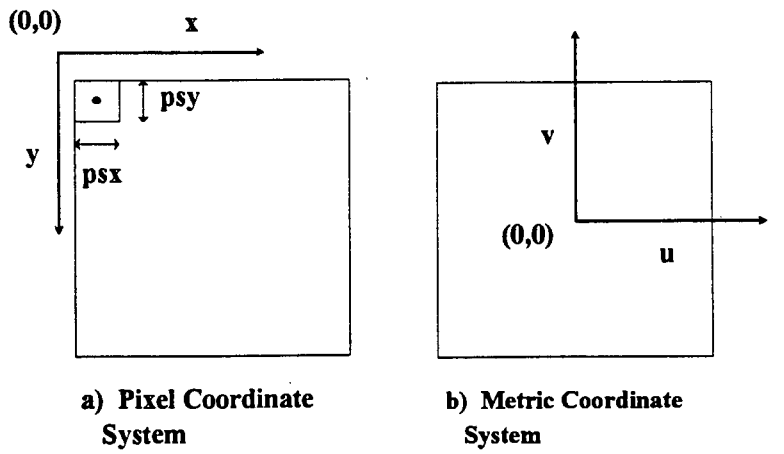


Figure 2.5: Pixel Image Coordinate System a) and Metric Coordinate System b)

With the advent of CCD sensors used in digital photogrammetry the accuracy of image measurements depends not only on characteristics such as the optics of the sensor system as in analog analytical photogrammetry, but also on the properties of the CCD camera itself. A detailed investigation by Lenz and Fritsch into some geometrical, optical and electronic

properties of CCD cameras in conjunction with analog/digital-converters and frame buffers can be found in [68]. A similar investigation is reported on by Beyer [11].

In this thesis the interior orientation was performed using a software package written in the Department of Surveying and Geodetic Engineering at UCT. The software is based on the bundle adjustment and uses known object points on a calibration frame to perform the bundle adjustment. Using the bundle adjustment for interior orientation is described in more detail by van der Vlugt [104] and Beyer [11].

2.3.2 Relative Orientation

The exterior orientation of a camera is defined by the position of the perspective centre in three-dimensional object space and the rotations of the image plane of the camera about the X,Y and Z axis. The optical axis of the camera is perpendicular to the image plane and the direction of this plane-normal describes the attitude of the camera at the time of exposure.

In traditional photogrammetry the relative orientation determines the position and attitude of the images in a stereo pair relative to each other. With the advent of multiphoto matching, as described by Baltsavias [7], the relative orientation is extended to more than two images. Refer to figure 2.6 for the geometric relationship between two cameras during relative orientation.

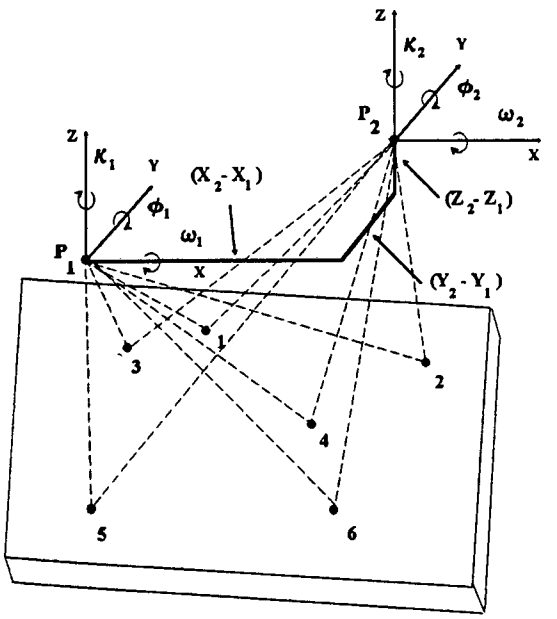


Figure 2.6: Image Geometry Used in Relative Orientation

The perspective centre P_1 for image 1 has object space coordinates (X_1, Y_1, Z_1) and the rotation angles about the X,Y and Z axes are labelled as ω_1, ϕ_1 and κ_1 respectively. Similarly the perspective centre for image 2 has object space coordinates (X_2, Y_2, Z_2) and the rotation angles about the X,Y and Z axes are labelled as ω_2, ϕ_2 and κ_2 respectively.

The vector $\vec{B}$ between the two perspective centres P_1 and P_2 defines the change in position

from one camera station to the other. The relative orientation parameters of interest are

$$\begin{aligned} B_X &= X_2 - X_1 \\ B_Y &= Y_2 - Y_1 \\ B_Z &= Z_2 - Z_1 \\ \Delta\omega &= \omega_2 - \omega_1 \\ \Delta\phi &= \phi_2 - \phi_1 \\ \Delta\kappa &= \kappa_2 - \kappa_1 \end{aligned}$$

The relative orientation between two images is known if two out of the three elements of the B-vector are known plus the three rotation angles. Generally the left perspective centre P_1 is chosen as the origin of the relative coordinate system with B_X chosen to control the scale of the model [105] [47].

A *normalized* image pair refers to an image pair which has been corrected for the relative orientation between the images, effectively transforming both images to the same plane. During normalization the right image is resampled in epipolar geometry, resulting in the epipolar lines falling along the scanlines of the digital image. In this dissertation the term *rectification* will be used synonymously with *normalization.*

In this thesis the relative orientation between the two images is automatically calculated from matched feature points. A minimum of five corresponding points are needed to yield a solution to the relative orientation. Generally more than five points are used to yield an overdetermined solution. A detailed explanation of the relative orientation calculation follows in chapter 7 later in this dissertation.

2.3.3 Absolute Orientation

During absolute orientation the stereo model is scaled, translated and rotated with respect to a known ground reference coordinate system [105]. In the mathematical sense this represents a seven parameter coordinate transformation from the object space of the stereo model to the object space of the ground reference coordinate system. Absolute orientation can either follow after relative orientation or can be calculated after the interior orientation. Refer to figure 2.7 for the system geometry in absolute orientation.

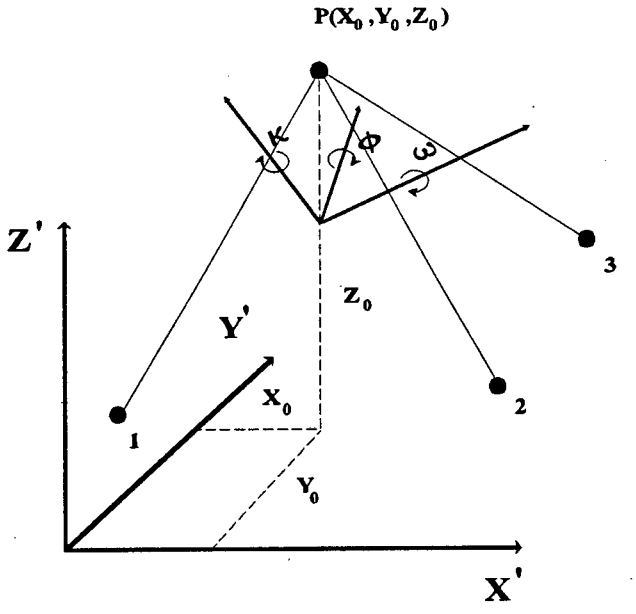


Figure 2.7: Image Geometry Used in Absolute Orientation

If X,Y and Z represent the coordinates of a point in the object space of the model and X',Y' and Z' represents the same point in the ground reference coordinate system then the transformation from one coordinate system to the other can be represented using the perspective projection transformation. i.e.

$$\begin{pmatrix} X' - X_0 \\ Y' - Y_0 \\ Z' - Z_0 \end{pmatrix} = \lambda \, R^T_{(\Omega,\Phi,K)} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{2.14}$$

The transformation parameters include the scale λ, three translation parameters (X_0, Y_0, Z_0) and three rotation parameters Ω,Φ and K. The rotation matrix $R_{(\Omega,\Phi,K)}$ is similar to the rotation matrix $R_{(\omega,\phi,\kappa)}$ defined in equation 2.2. The transformation parameters are solved for using ground control points with known ground coordinates, shown as points 1..3 in figure 2.7. Each control point gives rise to three equations, meaning that a minimum number of three control points are needed to solve for the seven coordinate transformation parameters.

In this thesis project the absolute orientation of the stereo model was never performed due to the nature of the application. In the close range work of industrial inspection the measurements needed are mostly of a relative nature which implies that only a relative orientation is needed. In the relative orientation the scale is relative and has to be solved for during industrial inspection. This problem is discussed in more detail in section 8.4.2.

Chapter 3

Fundamentals of Signal- and Image Processing

In digital photogrammetry, the pre-processing steps such as filtering and edge detection are based on digital image processing techniques. In this thesis one of the most critical parts of the matching scheme is *Feature Matching*. The feature matching candidates are found using a correlation function, which is calculated using a frequency domain representation of the particular feature. Filtering or correlation functions in digital image processing can be calculated in either the time or the frequency domain.

A digital image is a two dimensional representation of the light intensity as a function of the x and y spatial variables [85]. We introduce the concept of *spatial-time* where the x and y spatial variables are analogous to the time variable of a one-dimensional signal.

In Fourier Analysis, which is the fundamental transformation between the time and the frequency domains, the time-domain signal is represented as the sum of the Fourier components, represented by the coefficients of sines and cosines [81]. As the sine and cosine functions are periodic with period 2π, the *angular frequency* ω is introduced, where

$$\omega = \frac{2\pi}{t} \tag{3.1}$$

In a digital image Fourier Analysis can be used to transform from the *spatial-time* domain to the *spatial-frequency* domain [69]. The spatial frequency in this case will indicate the frequency distribution of the changes in greyvalue. If there are only gradual changes in greyscale over the image, the frequency components will mainly be contained in the lower spatial frequency band. If on the other hand there are a lot of abrupt greyscale changes or edges, the frequency components will be mainly contained in the higher spatial frequency band.

In this chapter some of the fundamentals of signal processing will be discussed. As we deal with digital images, which is a discrete-time representation of an analog image, emphasis will be put on discrete-time or digital signal processing techniques. The reader is referred to Oppenheim and Schafer [81], Lim [69] and Rosenfeld and Kak [85] for a more detailed

discussion of digital signal processing.

3.1 Continuous vs. Discrete Signals

A continuous signal can be defined as a mathematical function of one or more independent variables [81]. A speech signal for example represents the amplitude of the speech as a function of time, and a photographic image represents the light intensity as a function of the x and y spatial variables.

A continuous-time system refers to a signal processing system in which both the input signal and the output signal are continuous functions. Similarly, a discrete-time system refers to a signal processing system in which both the input signal and the output signal are discrete functions.

A discrete-time signal is formed by sampling a continuous-time signal at a set sampling time-interval T(figure 3.1). If the signal is $y(t)$, then the discrete-time signal $Y[n]$ can be defined as:

$$Y[n] = y(nT) \quad n = 0..N \tag{3.2}$$

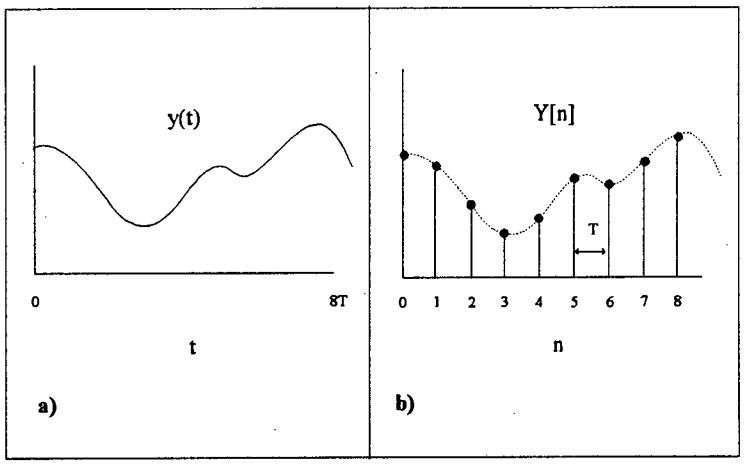


Figure 3.1: Forming a discrete signal Y[n] from a continuous signal y(t)

An image used in digital photogrammetry is either a scanned image from a photograph or an image captured by a digital CCD camera. It is thus a discrete-time representation of the image intensity as a function of the x- and y spatial variables.

Throughout this dissertation a continuous function will be indicated by round brackets, e.g. y(t), and a discrete function by square brackets, e.g. Y[n].

3.2 Frequency-Domain Representation of Signals

A continuous time signal $f(t)$ can be represented in the frequency-domain as a function of the *angular frequency* ω. The fundamental relation between the time-domain and frequency-domain representations of a signal is called the *Fourier Transform pair*, which is defined as follows:

$$\text{Synthesis:} \quad f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega \tag{3.3}$$

$$\text{Analysis:} \quad F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{3.4}$$

where the angular frequency ω is related to the frequency f by $\omega = 2\pi f$ and j is the same as the complex variable i where $i = \sqrt{-1}$. [1] Note that both the synthesis and analysis equations contain the complex exponentials $e^{j\omega t}$ where

$$e^{j\omega t} = cos(\omega t) + jsin(\omega t) \tag{3.5}$$

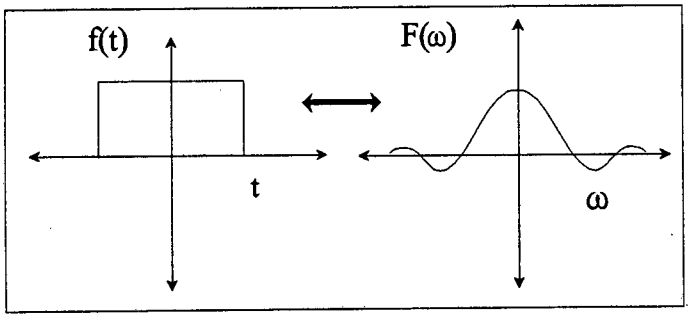


Figure 3.2: Fourier Transform pair $f(t) \Longleftrightarrow F(\omega)$

Equation 3.3 gives an expression for reconstruction of the signal $f(t)$ from the frequency domain representation $F(\omega)$ and is called the *synthesis* equation. Equation 3.4 gives an expression for analyzing the frequency components $F(\omega)$ of $f(t)$ and is called the *analysis* equation.

A Fourier Transform pair is usually indicated as

$$f(t) \Longleftrightarrow F(\omega)$$

The synthesis and analysis equations for discrete-time signals, linking the discrete time-domain sequence $f[n]$ to the continuous frequency domain function $F(\Omega)$ are defined as follows:

[1] In electrical engineering the symbol i is usually used to indicate *current*, therefore j is used instead to indicate a complex number.

$$f[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\Omega) e^{j\Omega n} d\Omega \tag{3.6}$$

and

$$F(\Omega) = \sum_{n=-\infty}^{\infty} f[n] e^{-j\Omega n} \tag{3.7}$$

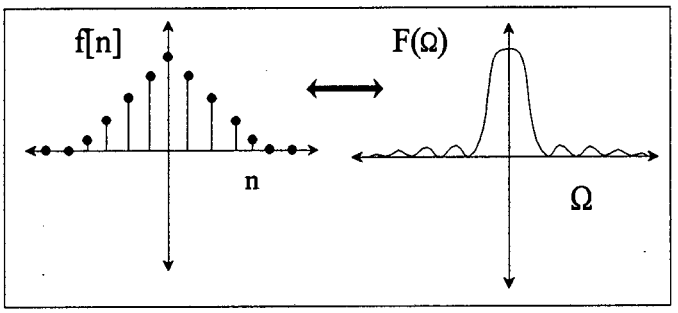


Figure 3.3: Fourier Transform pair $f[n] \Longleftrightarrow F(\Omega)$

The discrete function $f[n]$ need not be a function of "time", but it could for example be a function of "spatial-time" of an image. The function $f[n]$ could represent a one-dimensional section through a two-dimensional image. The "spatial-frequency" distribution of the grey-values will then be represented by the continuous frequency spectrum $F(\Omega)$. Note the capital Ω will be used throughout this dissertation to distinguish between the frequency spectrums of discrete and continuous functions.

A digital image is a discrete representation of a 2-D image intensity field captured by a CCD camera or obtained from scanning an analog image such as a photograph. If we label the samples along the x- and y axes as n_1 and n_2, we can form the Fourier Transform pair for this 2-dimensional signal $x[n_1, n_2]$ of size M x N as

$$x[n_1, n_2] = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(\omega_1, \omega_2) e^{j\omega_1 n_1} e^{j\omega_2 n_2} d\omega_1 d\omega_2 \tag{3.8}$$

and

$$X(\omega_1, \omega_2) = \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} x[n_1, n_2] e^{-j\omega_1 n_1} e^{-j\omega_2 n_2} \tag{3.9}$$

See figure 3.4 and figure 3.5 for typical examples of Fourier Transform pairs of 128x128 sub-images.

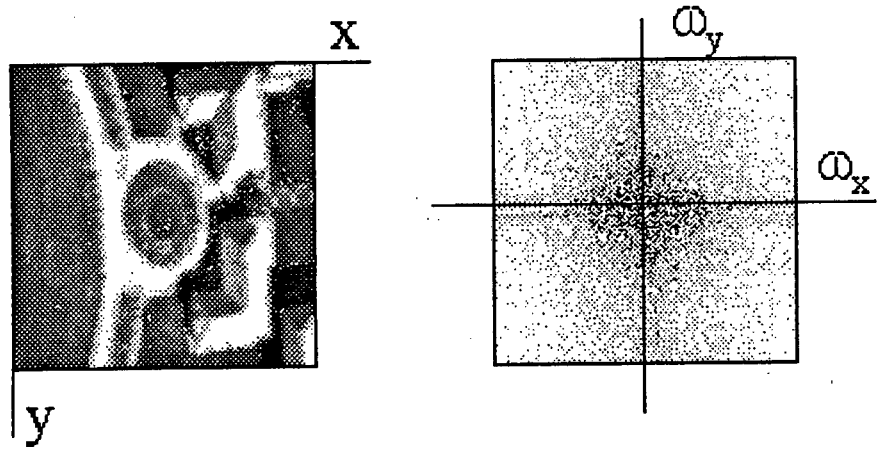


Figure 3.4: Spatial-Time a) and Spatial Frequency b) representation of an aerial sub-image

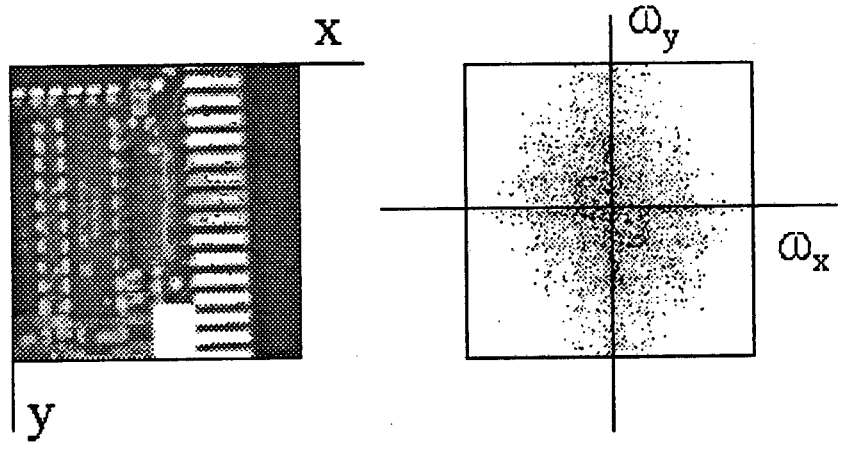


Figure 3.5: Spatial-Time a) and Spatial Frequency b) representation for a close-range sub-image of a PCB

Note in figure 3.5 the higher frequencies in the y direction are stronger due to the repetition of the black-to-white transitions in the y -direction.

In digital signal and image processing we deal with finite duration sequences, which allows us to form a finite duration *Discrete Fourier Transform* (DFT) which itself is a discrete-frequency sequence as opposed to the continuous function $F(\Omega)$. The DFT is thus a transformation from discrete-time to the discrete-frequency domain.

The Fourier Series representation of a continuous-time periodic signal generally requires infinitely many harmonically related complex exponentials, whereas the Fourier Series for any discrete-time signal with period N requires only N harmonically related complex exponentials [81].

This can be seen by examining the complex Fourier exponentials that form the basis of the Fourier series. For a discrete-time periodic sequence with period N, the Fourier exponentials

will be at integer multiples of the fundamental frequency $(2\pi/N)$. These periodic complex exponentials $e_k[n]$ are of the form

$$e_k[n] = e^{j(2\pi/N)kn} = e_k[n + rN] \tag{3.10}$$

where k, r and n are integers.

Due to the periodic nature of the function $e_k[n]$ over a frequency range of 2π which corresponds to N samples, we need only consider the first N values of this Fourier Series.

Consider a sequence $x[n]$ that is periodic with period N. The synthesis equation now has the form

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j(2\pi/N)kn} \tag{3.11}$$

and the analysis equation

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn} \tag{3.12}$$

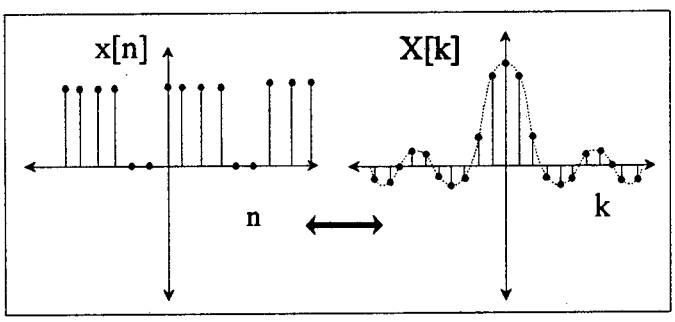


Figure 3.6: Fourier Transform pair $x[n] \Longleftrightarrow X[k]$ for a periodic signal

Research into the efficient computation of the Discrete Fourier Transform (DFT) has led to the development of the Fast Fourier Transform (FFT), which drastically reduces the number of computations that need to be done. The FFT algorithm is a *decimation-in-time* operator which sequentially divides the data streams in two to simplify the DFT calculation.

For the general case where N is an integral power of two i.e. $N = 2^v$ there will be v stages in the computation where $v = log_2 N$. For each stage we have N multiplications and additions, so the total number of computations gives a time-complexity of $N log_2 N$. If we compare this to the time-complexity of N^2 by the direct-computation from the analysis equation 3.12, we see that the decimation-in-time FFT computation is a lot more efficient. See figure 3.7

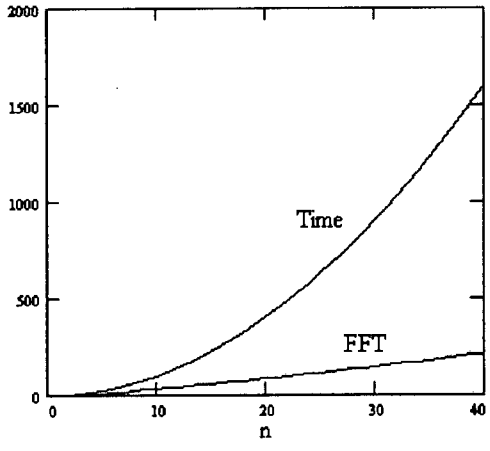


Figure 3.7: Time complexity comparison between the Time-domain and FFT computations of the DFT

The FFT of an image can be computed using this FFT algorithm to compute the FFT of each of the rows and columns individually and adding the outputs together.

3.3 Convolution

The image processing functions that form part of digital photogrammetry are mostly involved in the pre-processing of the images. The most commonly used pre-processing done on any image includes *Smoothing* and *Edge Enhancement*, which are effectively filtering operations.

These operations are performed by convolution of the image data with a smoothing or an edge enhancement function. In the 2-dimensional case such a discrete 2-D smoothing function is referred to as a *kernel*. A brief discussion of convolution in the 1-dimensional case will follow. For a more detailed discussion of 1-D convolution refer to Oppenheim and Schafer [81] and to Lim [69] and Rosenfeld and Kak [85] for 2-dimensional convolution.

3.3.1 Continuous-Time Convolution

If we have a time-dependent signal $f(t)$, and convolve it with a filter that has an impulse-response $h(t)$, the resultant signal is $y(t)$, formed by the *convolution integral*

$$y(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \tag{3.13}$$

This forms a means of linking the output signal $y(t)$ to the input $f(t)$ and is denoted $y(t) = f(t) \otimes h(t)$ where $\otimes$ means convolution.

If $f(t)$ has a Fourier pair $F(\omega)$ and $h(t)$ has a Fourier pair $H(\omega)$, the output $y(t)$ will have a Fourier pair

$$Y(\omega) = F(\omega)H(\omega) \tag{3.14}$$

This is a very important result which states that convolution in the time-domain is equivalent to multiplication in the frequency-domain. The convolution output $y(t)$ can thus be obtained from $Y(\omega)$ using the synthesis equation 3.3 From this result it is clear why filter-design is mainly done in the frequency-domain, as the resultant output frequency spectrum can be predicted.

3.3.2 Discrete-Time Convolution

For a discrete signal $x[n]$ we can form the *convolution-sum* , which is similar to the convolution-integral for the continuous case. The convolution of input signal $x[n]$ with a filter with impulse-response $h[n]$ gives result $y[n]$ where

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{3.15}$$

and the convolution-sum is denoted as $x[n] \otimes h[n]$.

As for the continuous case, if $x[n]$ has frequency-spectrum $X(\Omega)$ and $h[n]$ has frequency-spectrum $H(\Omega)$ then the resultant frequency-spectrum $Y(\Omega)$ of $y[n]$ will be formed by the product of the two frequency spectrums

$$Y(\Omega) = X(\Omega)H(\Omega) \tag{3.16}$$

As previously the convolution output can be obtained from the Inverse Fourier Transform of $Y(\Omega)$.

In image processing convolution is typically performed using a 2-D kernel as smoothing filter or edge-enhancement function. The 2-dimensional convolution between an image $x[n_1, n_2]$ with a kernel $h[n_1, n_2]$ is defined as

$$y[n_1, n_2] = x[n_1, n_2] \otimes h[n_1, n_2] \tag{3.17}$$

$$= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x[k_1, k_2]h[n_1 - k_1, n_2 - k_2] \tag{3.18}$$

As in the 1-dimensional case, if $x[n_1, n_2]$ has Fourier-spectrum $X(\omega_1, \omega_2)$ and $h[n_1, n_2]$ has Fourier-spectrum $H(\omega_1, \omega_2)$, the output Fourier spectrum $Y(\omega_1, \omega_2)$ is

$$Y(\omega_1, \omega_2) = X(\omega_1, \omega_2)H(\omega_1, \omega_2) \tag{3.19}$$

The convolution output $y[n_1, n_2]$ can now be obtained from the Inverse Fourier Transform of equation 3.8.

In this thesis project the edge detection step is performed by the convolution of a discrete-time edge enhancement operator with the image data. During the feature matching step, a correlation function between image features is calculated using FFT's.

Chapter 4

Feature Extraction

The extraction of significant features in an image is the first and most important step in image analysis. Image analysis forms the link between the image data and the "real world", and feature extraction focuses on locating and interpreting perceptual structures that can be linked to a model of the "real world".

In the literature many feature extraction techniques have been reported on. Förstner distinguishes between three types of feature extraction [25]:

- Low Level Features: Attributes of the radiometric data such as texture.
- Mid Level Features: Geometric structures such as points, lines and regions.
- High Level Features: Complex combinations of mid level features that have already been interpreted such as rooftops, buildings and roads.

The extraction of low level spectral features is used in the area of Remote Sensing and will not be discussed in this dissertation.

Of greatest interest in digital photogrammetry is the extraction of mid level features, which represents the geometric features in an image [42] [25]. A common trend in feature extraction is to look at not only the spatial information of a feature but also the spectral and topographic information of a feature [101].

Point Features Of the more well known point operators used in digital photogrammetry are the *Moravec* operator [42] [30] and the *Förstner* operator [26] [25]. The Förstner operator detects distinct points such as corners and centres of circular features. Features are detected using a two-step procedure which first searches for an optimal window and then estimates an optimal position for the feature within the window. The reader is referred to section 5.3 for a more detailed description of the Förstner Interest Operator.

Template Matching techniques exist to find distinct points used in image analysis. For example, Singh and Shneier employ template matching to find corners in images [93] and Gruen

and Stallmann use synthetic edge templates to track significant edges [41].

Corners can also be found by analyzing one-dimensional descriptors of two-dimensional shapes. Freeman and Davis report on a corner finding algorithm that detects changes in chain coded curves and produces a measure of the "cornerity" of a corner [27]. Gottschalk and Mudge devise a set of features called scale-invariant critical point neighbourhoods (SICPN's) which efficiently encode the local shape of edge segments near points of high curvature. The $\phi(s)$ plot of a feature contour is normalized and the SICPN's are obtained by resampling the $\phi(s)$ plot in a small interval around critical points of high curvature in the normalized $\phi(s)$ plot [32]. In this thesis project corners are detected by searching for local maxima in the first derivative of the $\phi(s)$ plots [103].

Line Features Lines in an image represent the boundaries between for example a foreground structure such as a house and the background. In image analysis lines are extracted using edge detection followed by line extraction. Edge detectors are based on the greyvalues in an image, and detect edges where there is a large change in greyvalues. Detected lines in an image do not always coincide with a physical boundary of a structure such as a house. Examples of spurious edges are edges due to shadow effects or edges due to differences in reflective properties of the scene, which also result in a large change in greylevels. Figure 4.1 shows two examples of aerial sub-images with the resultant edge-maps found through edge detection. In the example at the top the edges found represent the physical boundaries of the circular feature and the houses. In the second example the shadow of the house is clearly visible in the top left of the sub-image, which leads to an edge in the edge map which does not represent the physical boundary of the house but actually is a "ghost image" of this boundary.

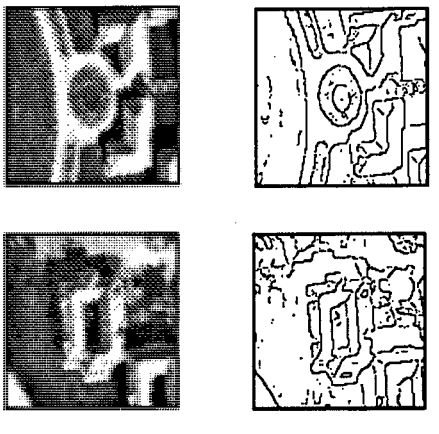


Figure 4.1: Aerial sub-images with edge features

Of specific interest in image analysis is the detection and interpretation of cultural (man-made) features specifically in aerial images. Cultural features are found by searching for well defined geometric shapes such as straight lines, rectangles and circles. The *Hough Transform* is a well known operator which can be used to find features such as straight lines and circular arcs [47] [58]. Huertas has done significant work in detecting cultural features from aerial imagery. Huertas and Nevatia describe a method for detecting buildings in aerial images [57].

They first detect line segments through edge detection followed by line following. Corners that are close to orthogonal are detected and then labelled based on shadows. Finally object boundaries are traced and hypotheses are verified. In a paper by Huertas, Cole and Nevatia [55], a similar approach is used to detect runways in complex airport scenes by looking for instances of long rectangular shapes. Price and Huertas use *perceptual grouping* techniques to detect cultural features in aerial images [83]. Perceptual organization refers to the ability of a visual system to quickly capture visual representations of structure and similarity among otherwise random elements, features and patterns. Even though the image analysis methods proposed by Huertas *et al* were applied here to single images, a similar approach can be taken for stereo vision.

He and Novak present a road analysis system that recognizes road edges and centre lines by extracting straight line segments in a stereo pair [48]. The extraction of parallel straight lines is also used to recognize mile markers along the road by tracing vertical edges in the image. Many examples of line features used in image matching schemes will be presented in chapter 6 on feature matching.

In recent years the extraction of line features has not been limited to line features in image-space. Strunz matches straight lines in object space and determines the relationships between points, lines and surfaces in object-space and their projections in image-space [97]. Krupnik and Schenk do segmentation of images in 3-D object space [65] after transforming 2-D curves to 3-D object space. Straight lines and regular curves are segmented in object space using methods such as split-and-merge and a 3-D version of the $\phi(s)$ method.

In this thesis lines are extracted and classified according to shape. Of specific interest are closed line features that form polygons such as rectangles, circles and other well defined polygon-features. An open line feature will be referred to as an ARC and a closed line feature as a POLYGON.

Region Features Regions of interest are used more in Remote Sensing to isolate regions with e.g. similar spectral properties. Note that here low-level features such as texture play a significant part in distinguishing one region from another. The method of distinguishing one region from another is generally referred to as *segmentation.* Segmentation tries to group piecewise homogeneous areas in an image together to distinguish between different types of regions . Some of the well known segmentation techniques are *Thresholding* [46] [63], *Region Growing* [85] [46], *Spatial Clustering* and *Split-and-Merge* [46]. Rosenfeld describes the *Relaxation* method for doing iterative segmentation [85]. The relaxation technique starts off by making probabilistic classification decisions and then at each iteration adjusting these decisions by examining the probabilistic classifications of the surrounding points made at the previous iteration.

Interest regions used in digital photogrammetry include "interest clumps" as described by Fuller and Ehlers [30], which are connected regions with interest values above a certain threshold . Interest values used by Fuller and Ehlers are obtained through using the Moravec operator. Gülch extracts grey level "blobs" and traces them through scale-space. The significance of these scale-space blobs is estimated by tracing their volumes through scale-space and sorting the blobs in descending order of significance.

Using high-level features is becoming more prominent in image analysis. To interpret high-level features knowledge about not only the geometric features but also the relations between features is needed. Significant research in this field has been reported on by Braun [13] . Line segments are grouped together in image space and then different entities such as lines and points are grouped together in 3-D object space. The relations and geometric properties between the 3-D entities grouped together is used to find instances of 3-D structures such as edges, vertices and surfaces such as rooftops. Buildings are simplified as being polyhedra with groups of parallel and orthogonal lines and rectangular corners. Related to the work of Braun is the work of Schickler [90]. A 3-D wireframe model is used to automatically perform the exterior orientation of a single image. Straight line segments are extracted and an attempt is made to match straight line segments to the 3-D wireframe model of a building.

Gülch uses a Knowledge Base that includes factual, strategic and judgemental knowledge to do parsing on aerial images of buildings. Classical feature extraction techniques like arc extraction are used in conjunction with rules such as general rules, factual rules about buildings and rules based on heuristic experience contained in the knowledge base. Buildings are described in 3-D object space according to the structure of the building in terms of the orientation of line segments and the spatial and contextual relationship between line segments such as vertices being formed where two or more line segments meet [43].

In the work of Huertas *et al* [57] [55], the authors attempt to not only group lines and corners together but also to classify polygons as being a "runway" or a "building", thereby attaching some meaning and higher order knowledge to the rectangular features.

A model-based image analysis system is introduced by Maitre and Luo [73]. The authors assume that an observed 3-D scene is made of different objects, where each is composed of multiple surfaces of which a significant number may be covered by parametric functions such as planar or quadratic surfaces. The best functions belonging to each object surface is found by observing the disparity maps for an image. Disparity in a stereo pair refers to the difference in positions of the images of the same three-dimensional point in two perspective projection images taken from different positions in object space [47].

In this project line features are extracted and labelled either as an ARC (open line segment) or a POLYGON (closed line segment). During feature classification a decision is made as to the type of polygon i.e. how many corners there are or if a polygon is a circle. Lines are extracted using edge detection followed by line following. In this thesis project the Canny edge detector is used. To ensure that polygon features can be matched some basic editing on the edge image is allowed before features are extracted through line following. Line features are vectorized by storing the (x, y) boundary positions of each edge pixel of the line feature in an array. The vectorized feature boundaries are finally refined to the sub-pixel level using a sub-pixel edge detector, the "preservation of moments" edge detector.

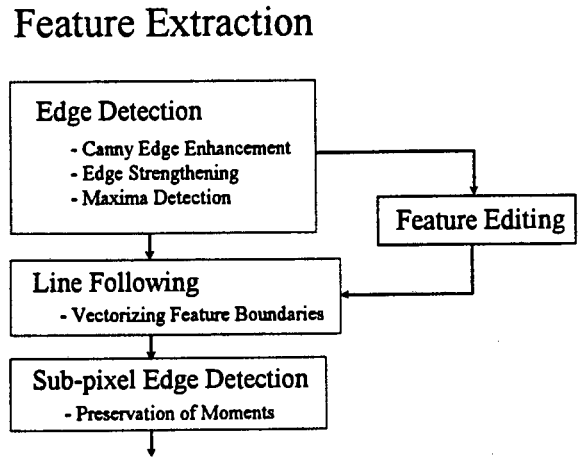


Figure 4.2: Feature Extraction Flowchart

In this chapter some edge detection techniques will be described and compared according to their performance and complexity. Line following will described next, followed by a brief discussion of the feature editing procedure.

4.1 Edge Detection

4.1.1 Introduction

Edge detection is used to find the boundaries of features in an image. The boundary of a feature is represented by a sharp transition in the greyvalues. An edge detector finds this transition and thus marks the feature boundary. A large difference in greyscales between two adjacent pixels will represent a strong edge and a small difference in greyscale will represent a weak edge.

As an example of edge detection an image of a Printed Circuit Board (PCB) is shown in figure 4.3. The Integrated Circuits (IC's) on the PCB are painted white to clearly define the rectangles representing those IC's. Without painting the tops of the IC's the contrast between the IC boundary and its surrounds is not enough to clearly define the outer boundaries of the IC's.

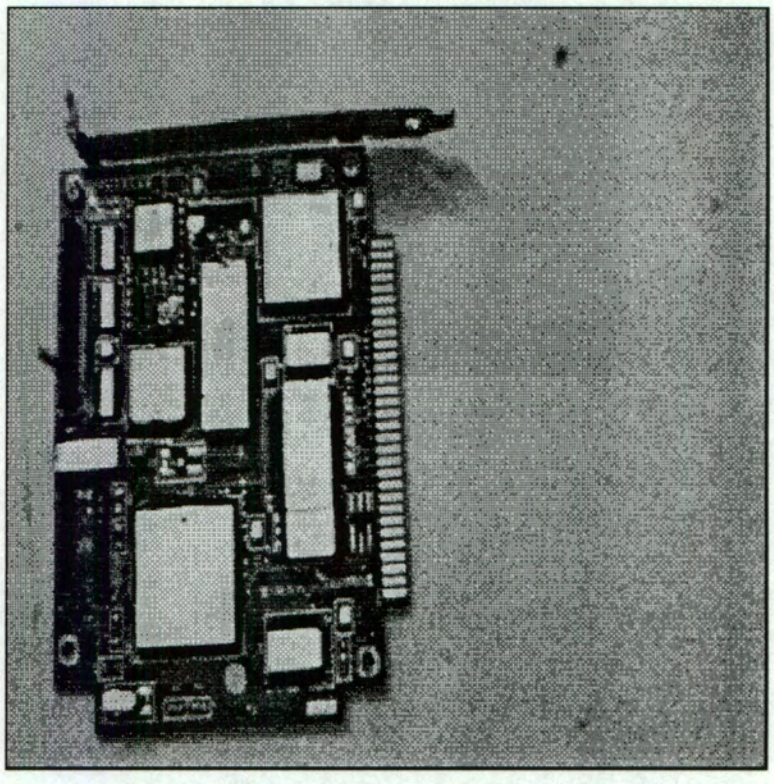


Figure 4.3: Test PCB image with IC rectangles clearly defined

4.1.2 Edge Enhancement

The first step towards edge detection is edge enhancement on the raw data. In the literature, attention is given to creating an accurate model for step- and ramp edges [74] [45] [71]. Once these models have been defined, a one dimensional operator is designed which enables accurate detection of the simulated edge. This operator and definition of the edge can be extended into the two-dimensional image domain. The most common models are based on the first- and second derivatives of the signal [47] [85].

The most common edges found in digital images are step- and ramp edges. A 3-D profile of a step edge is shown in figure 4.4 a). For a step edge the greylevels change abruptly from one element to another, in this case from 0 to 200. The more common edges found in digital photogrammetry are ramp edges, which change more gradually from one greylevel to another than step edges. Figure 4.4 b) shows an example of a ramp edge with the edge profile changing in greylevels from 0 to 210 in the "transition band" of 5 pixels.

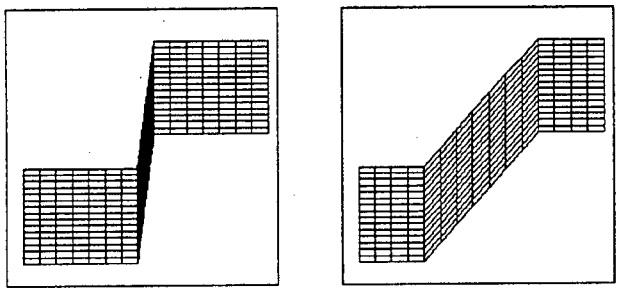


Figure 4.4: Examples of a step edge a) and a ramp edge b)

The first derivative of a digital signal gives the greyscale gradient of the signal. An edge is found where this gradient obtains a local maximum in a specific direction or small neighbourhood. The second derivative based methods such as the *Laplacian of Gaussian (LoG)* find edges by detecting zero crossings in 3x3 neighbourhoods of the operator response. Edge detection using the Laplacian of Gaussian function can be seen to be analogous to edge detection using the first derivative, as the maxima of the first derivative response are located at the zero crossings of the second derivative response. Of the first derivative based methods reported the *Canny* and "*Gradient-Sum*" methods have been implemented in this project and their performance analyzed. The only second derivative based method implemented in this project was the *LoG*. All the gradient based methods mentioned above were limited to pixel-accuracy , and were therefore used to define a small search region in which the edge can be found to sub-pixel accuracy in a subsequent step.

The edge detectors evaluated during this project are tabulated below:

Name	*Type*
Gradient-Sum	1_{st} derivative
Canny	1_{st} derivative,smoothing
Laplacian of Gaussian (LoG)	2_{nd} derivative,smoothing
Preservation of Moments	Statistics

Other methods using statistical analysis of the edges and frequency-domain analysis have also been reported in the literature [72] [59] [92] [78]. The reader is referred to [71] for a comparison of the performance of a wide range of edge detectors.

The method using the preservation of moments as proposed by Tabatabai et Al [98] has been implemented , giving sub-pixel measures for the edge location. More recent methods of edge enhancement have used "heat flow" around an edge and have been found to give a good response at corners and line crossings , two areas that have been found to be problematic , especially using linear edge detectors [79].

Local Gradient

One of the earliest gradient-based edge operators is the one proposed by Roberts(1965) [47]. He used two 2x2 convolution masks to calculate the gradient in two diagonal directions (see figure 4.5). Let g_1 be the convolution output of the first mask and g_2 from the second mask then the gradient magnitude is calculated as $g = \sqrt{g_1^2 + g_2^2}$

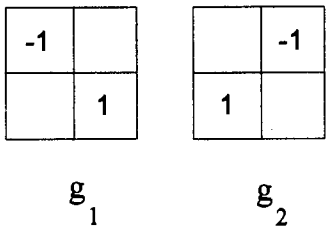


Figure 4.5: Roberts Edge Detector Masks

Prewitt(1970) [47] used two 3x3 convolution masks to calculate the local gradient (see figure 4.6). Let Let g_1 be the convolution output of the first mask and g_2 from the second mask then the gradient magnitude is calculated as $g = \sqrt{g_1^2 + g_2^2}$ and the gradient direction θ taken in a clockwise angle with respect to the x-axis is $\theta = tan^{-1}(g_1/g_2)$

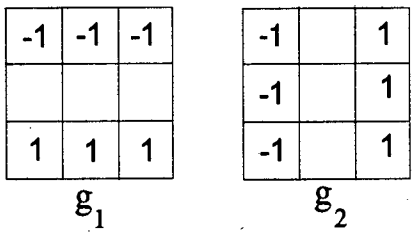


Figure 4.6: Prewitt Edge Detector Masks

Similar 3x3 convolution masks were employed by Sobel(1970) and Frei and Chen(1977) [47]. The only differences occur in the weighting of the central pixels in each row and column (see figure 4.7).

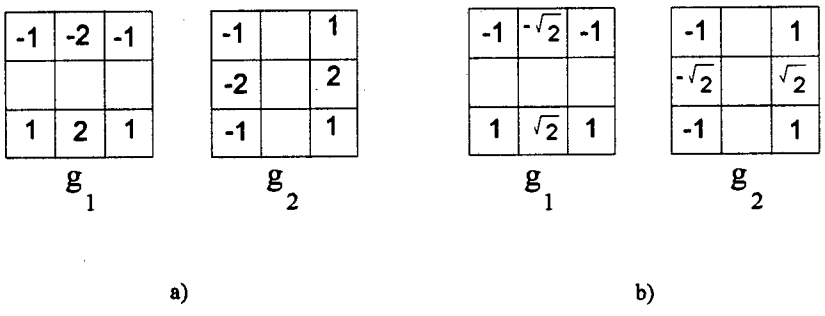


Figure 4.7: a) Sobel Edge Detector Masks and b) Frei and Chen Masks

The local gradient of a signal is calculated from the first derivative at a point in a specific

Type	*derivation*
backward difference	$g_i = f_i - f_{i-1}$
forward difference	$g_i = f_{i+1} - f_i$
average difference	$g_i = \frac{1}{2}(f_{i+1} - f_{i-1})$

direction. In a digital signal this can be calculated using the *backward difference forward difference* or *average difference* at a point i.e.

where g_i is the greyscale gradient at point i and f_i is the greyscale value at point i.

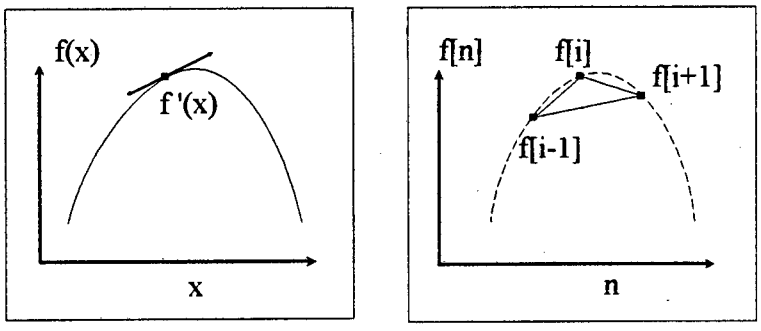


Figure 4.8: Determination of gradients for a) a continuous function and b) a discrete function

The method found to be more effective than the local gradient method is an extension of this method as proposed by Rosenfeld [85] , which takes into account the sum of the gradients on both sides of the pixel i.e. $g_i = (f_{i+2} + f_{i+1}) - (f_{i-1} + f_{i-2})$. The local gradient produces a large response for a large greyscale gradient, where the "Gradient-sum" is more immune to large edge spikes due to the smoothing effect of the summation. A standard practice when using the local gradient operator is to perform some smoothing on the image first before doing edge enhancement. The image is smoothed using a Low-Pass Filtering kernel such as an averaging filter or a Gaussian smoothing function.

The gradient sum operator $b(x)$ over any width n can mathematically be defined as:

$$b(x) = \begin{cases} 0 & \text{if } x < -n/2 \\ -1 & \text{if } -n/2 \leq x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } 0 < x \leq n/2 \\ 0 & \text{if } x > n/2 \end{cases} \tag{4.1}$$

where x is an integer denoting the greyscale sample and n is the width of the gradient operator.

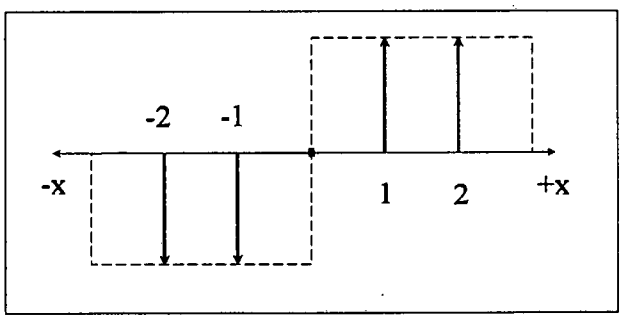


Figure 4.9: Gradient-sum box operator $b(x)$ for $n = 5$

From equation 4.1 it is simple to deduce that for a digital signal with n=3 the box-operator reverts to the local average gradient operator i.e. the elements of $b(x)$ will be $(-1, 0, 1)$ corresponding to $g_i = f_{i+1} - f_{i-1}$. In this thesis project an operator with width n=5 was chosen as suggested by Rosenfeld. The elements of the box operator $b(x)$ will thus be $(-1, -1, 0, 1, 1)$.

Figure 4.10 shows the resultant edge enhanced map from the convolution of the the left PCB image with the gradient-sum box operator with $n = 5$.

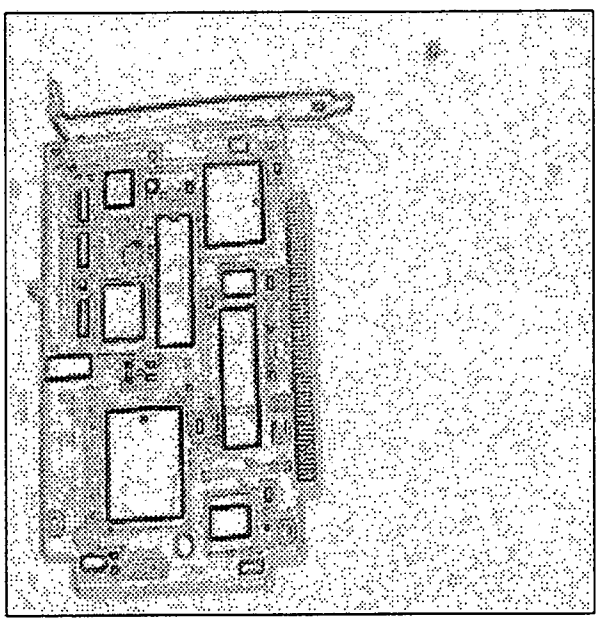


Figure 4.10: Gradient-sum edge enhanced map for the LEFT PCB image

See figure 4.11 for the frequency response of the Gradient sum box operator with n=5 in the frequency range $-\pi ... \pi$. The frequency response shows a band-pass characteristic, which can be viewed as the combination of a low-pass filter with a high-pass filter. The low-pass smoothing is due to the summation of the greylevels and the high-pass response is due to the subtraction of the gradient sums.

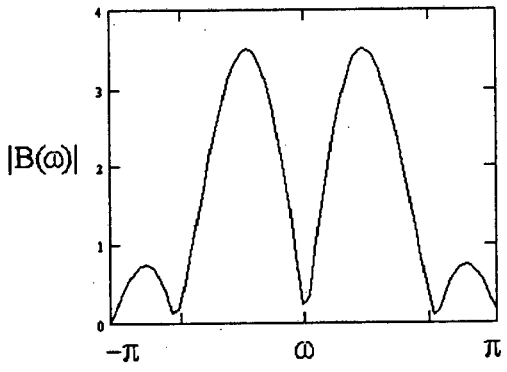


Figure 4.11: Frequency Response of the Gradient-box Edge Detector with n=5

In comparison we look at the frequency response of the Gradient-sum box operator with n=7 in figure 4.12. Note that the passband is narrower because of the extra smoothing from taking a wider kernel. The price paid for this extra smoothing can be seen from the side-lobes in the high frequency range, which should ideally be closer to zero. High frequency values usually represent unwanted "noisy" edges and the edge filter should be designed to suppress these unwanted edges.

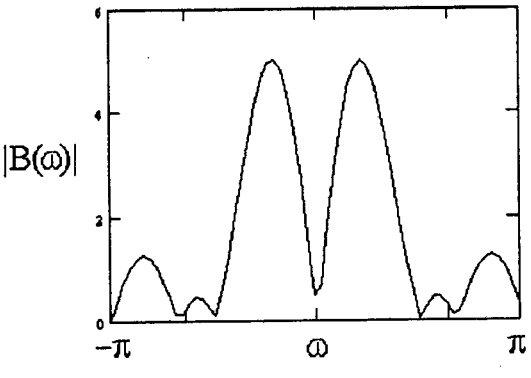


Figure 4.12: Frequency Response of the Gradient-box Edge Detector with n=7

In this project the method used by Rosenfeld gave a good response for reasonably clean edges. Taking the sum of adjacent gradients did perform some smoothing , and was considered to be sufficient for this application. The operator was not extended to two dimensions, but instead the one dimensional operator response was calculated in each of four possible edge-normal directions in a 5x5 neighbourhood as indicated in figure 4.13.

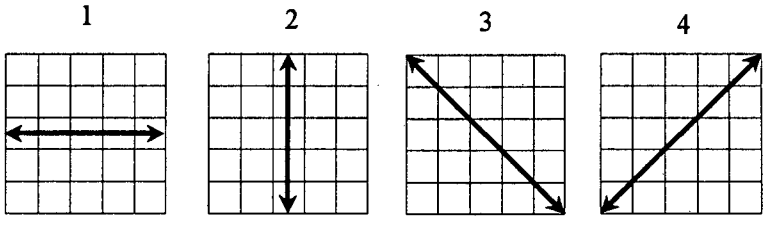


Figure 4.13: Major edge directions in a 5x5 neighbourhood

Note that the directions indicated will give a maximum response when the direction is normal to the edge-direction i.e. an edge that is located in direction (4) will give a maximum response in direction (3).

The resultant edge strengths are calculated from the convolution of the one-dimensional box-operator in each of the four indicated directions. Only the maximum of the four resultants is stored and the edge normal taken in the direction in which the maximum occurs.

Consider the example of the box operator h[n] being convolved with a symmetric step edge that is defined by:

$$X[n] = \begin{cases} 0 & \text{if } n < 0 \\ 0.5 & \text{if } n = 0 \\ 1 & \text{if } n > 0 \end{cases} \tag{4.2}$$

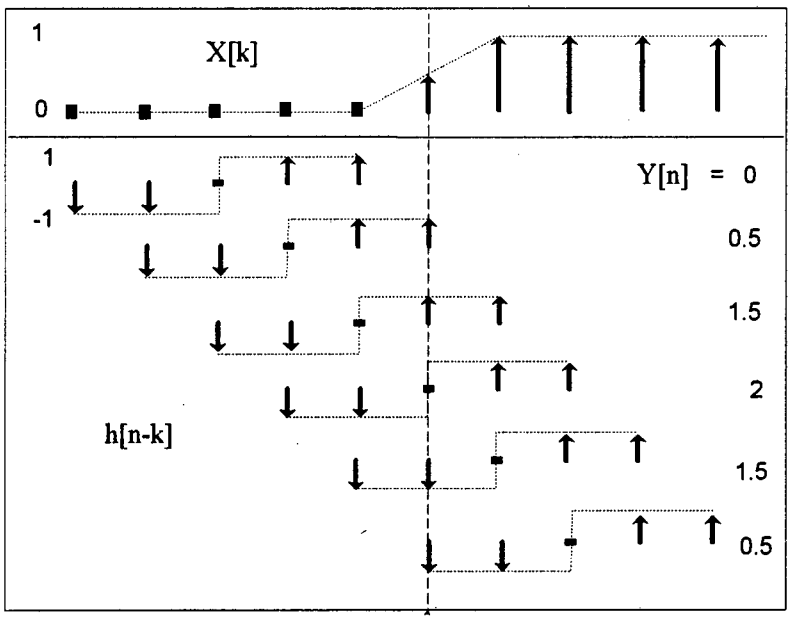


Figure 4.14: Convolution of the Gradient edge operator with a step edge

The convolution process is depicted graphically in 4.14 above. We recall from equation 3.15 that

$$Y[n] = \sum_{k=-\infty}^{\infty} X[k]h[n-k]$$

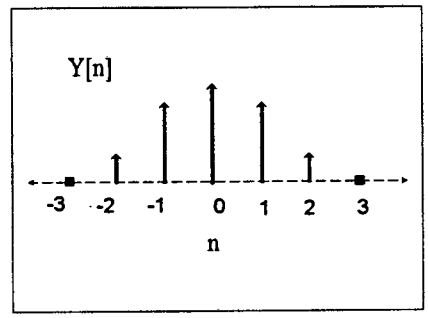


Figure 4.15: Resultant Y[n] of Graphical Convolution

The convolution output is zero until $[n-k]$ = -2. Here $Y[-2] = 0.5$, corresponding to $X[0]*h[2]$ = $(0.5 * 1)$. $Y[-1]$ is calculated as $(X[0] * h[1] + X[1] * h[2]) = (0.5 * 1 + 1 * 1) = 1.5$ This continues until $Y[2] = 0.5$, after which the output $Y[n] = 0$.

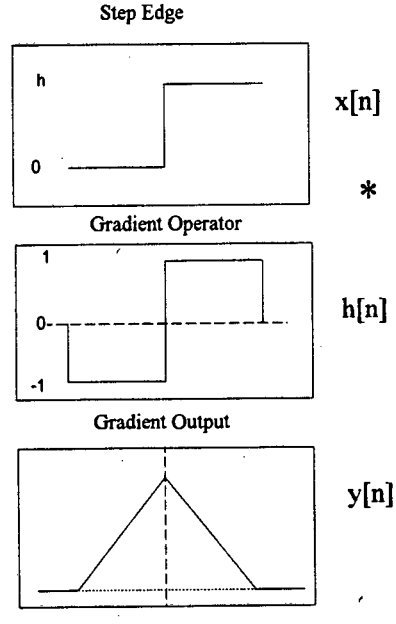


Figure 4.16: Convolution of the Gradient edge operator with a step edge

Refer to 4.16 for an example of the convolution of a box-operator with response b[n] with an ideal step edge x[n]. The output convolution-sum y[n] is indicated by the "Gradient Output".

Canny Edge Operator

The edge operator as first implemented by Canny(1986) [15] is based on the linear gradient of the input signal , with Gaussian smoothing as an integral part of the edge operator. Instead of first filtering the image and then finding the gradients, which is generally the approach during edge detection [74], the Canny operator combines the gradient with a Gaussian smoothing function. The level of smoothing is determined by the σ of the Gaussian function.

The one-dimensional Gaussian function is given by:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} exp \frac{-x^2}{2\sigma^2} \tag{4.3}$$

and its first derivative :

$$G'(x) = \frac{-x}{\sigma^2} G(x) \tag{4.4}$$

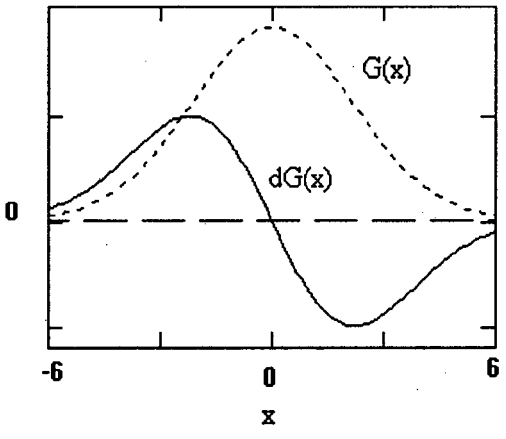
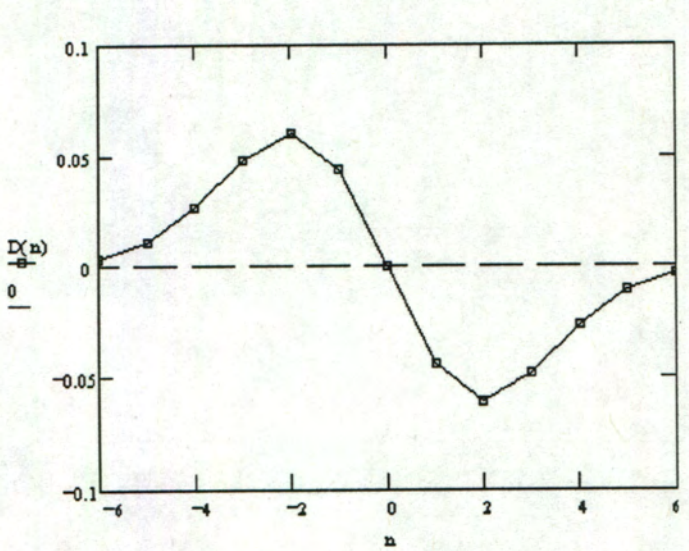


Figure 4.17: Gaussian G(x) and First Derivative dG(x)

The discrete nature of the data and the processing means that the continuous function $G'(x)$ has to be sampled at integer locations of x. The discrete function representing the impulse-response of a filter is generally referred to as the *convolution kernel.* In this thesis project the Canny convolution kernel is obtained by first calculating the width of the kernel, which is related to the amount of smoothing. Frequency domain analysis reported on by Sotak and Boyer [94] suggests a convolution kernel width of 6σ. The influence of the kernel width is reported on by Chen *et al* [16], Sotak and Boyer [94] and Grimson and Hildreth [37]. These authors analyze the effect of the kernel width for the Second derivative or Laplacian of Gaussian operator, but exactly the same principles apply for the First derivative of the Gaussian function.

Figure 4.18 shows an example of a discrete Canny filter with $\sigma = 2$. The width of the operator is calculated as the next odd number above 6σ i.e. 13. The width of the discrete filter is an odd number to preserve the symmetry around the zero crossing of the first derivative of the Gaussian. An increase in σ will thus increase the width of the filter and sample the continuous

function $G'(x)$ at different places resulting in different values of the kernel $D[n]$, shown in the table in figure 4.18.



D(n)
0.003
0.011
0.027
0.049
0.06
0.044
0
- 0.044
- 0.06
- 0.049
- 0.027
- 0.011
- 0.003

Figure 4.18: Discrete Canny kernel for $\sigma = 2$ and width $= 13$

Convolution of the discrete Canny kernel $D[n]$ with the image in each of the four directions as indicated in figure 4.13 is performed. The absolute maximum of these four convolutions is again taken as edge strength and the normal to the edge direction as the direction in which this maximum occurs.

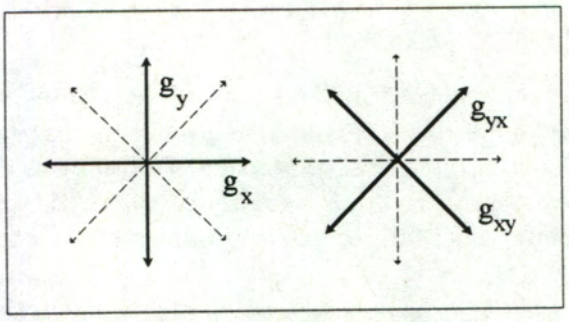


Figure 4.19: Basic gradient directions g_x and g_y

Figure 4.20 shows the resultant edge enhanced map from the convolution of the the right PCB image with the Canny edge operator with $\sigma = 2.0$.

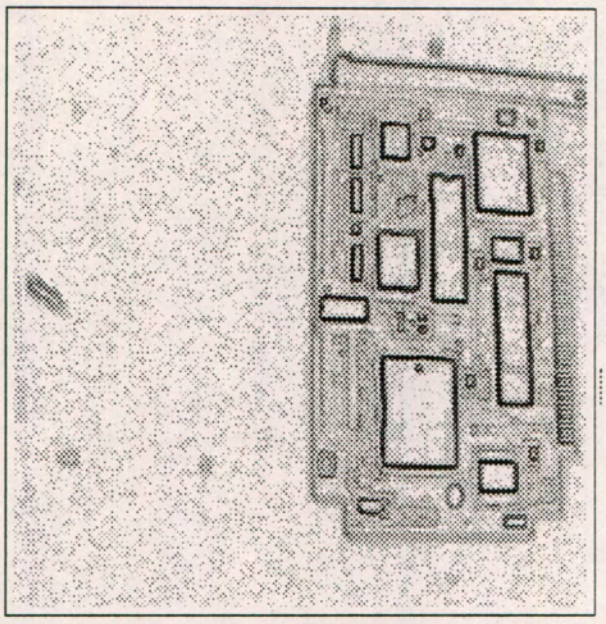


Figure 4.20: Canny edge enhanced map for the RIGHT PCB image

Note that with both the gradient-sum and the Canny edge operators the edge strength and edge directions can be obtained from only the x-gradient g_x and the y-gradient g_y, shown in figure 4.19.

The resultant edge strength g is now

$$g = \sqrt{g_x^2 + g_y^2} \tag{4.5}$$

with edge normal direction

$$\theta = tan^{-1}\frac{g_y}{g_x} \tag{4.6}$$

This edge normal direction θ is used in the sub-pixel edge detection described later in this chapter and the resultant edge strength is used as part of the feature description discussed in a later chapter. Experimentation showed that using the results of all four gradient directions shown in figure 4.19 gave a more accurate binary edge map with less gaps in the edge-chains, especially at corner points which are of extreme importance to the whole image matching scheme.

Figure 4.21 b) shows the plot of the edge normal directions in the range 0 to 2π scaled to a greyscale between 0 and 255 for the sub-image shown in figure 4.21 a). Note that the edge normal increases or decreases as we traverse the boundary of the feature. The edge normals for the inner and outer circles in the top left of the sub-image are in the opposite sense due to the difference in the edge directions, resulting in a difference of π degrees between the inner and outer direction plot. The outer circle has greyscale transitions from white to black, where the inner circle has greyscale transitions from black to white.

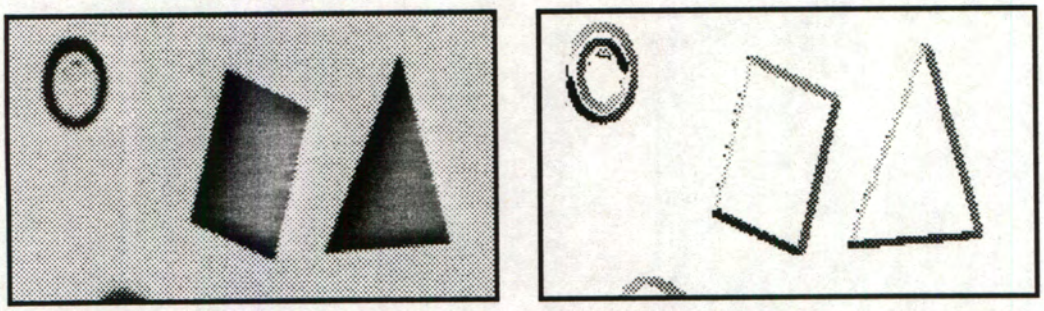


Figure 4.21: Inverted sub-image of objects a) and Resultant edge normal plot scaled from 0 to 255

Figure 4.22 shows the frequency response of the Canny edge detector. It has the same bandpass characteristic as the Gradient-sum box operator, but the frequency response of the Canny edge detector is better as it has a narrower passband than the Gradient-sum box operator and higher attenuation of the higher frequencies. A kernel width of 13 has been used to ensure a good frequency domain response.

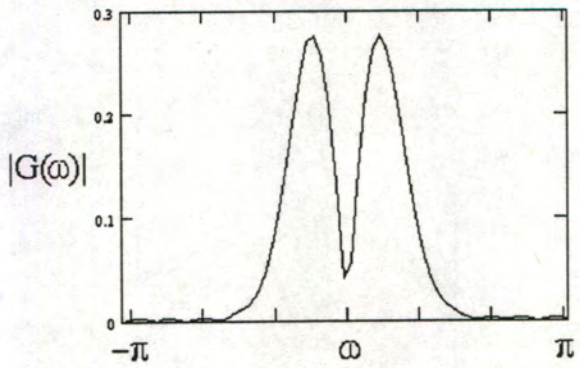


Figure 4.22: Frequency Response of the Canny edge operator with $\sigma = 2.2$ and width $= 13$

In comparison we observe the frequency domain response of a Canny edge operator of width 13 and σ of 5. The width of the kernel is not sufficient to ensure a good frequency response, which can be seen from the large ripples in the higher frequencies displayed in figure 4.23. A width of 6σ would indicate a kernel width of 31, more than twice the size of the kernel width shown. The larger value of sigma however means more smoothing, which is represented by the narrower main peak in the passband.

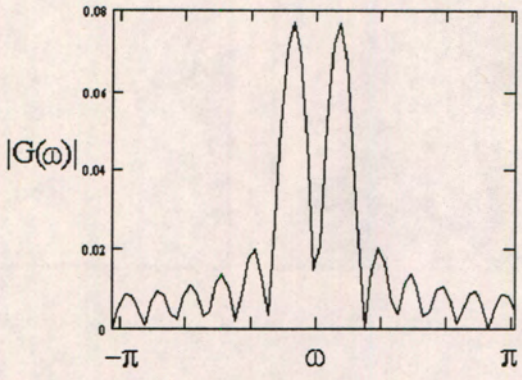


Figure 4.23: Frequency Response of the Canny edge operator with $\sigma = 5$ and width = 13

The results of the Canny edge operator can be combined by *cascading* Canny edge operators of different widths. When different filters are used sequentially on a signal and the output is combined we refer to the filters as being cascaded. The merits of cascading Canny and other edge detection filters of different widths is discussed in more detail by Canny [15], Bergholm [10], Lacroix [67] and Jeong and Kim [60].

The Gaussian function can be extended to two dimensions as:

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} exp \frac{-x^2+y^2}{2\sigma^2} \tag{4.7}$$

assuming that the function has the same σ in both the x and y directions.

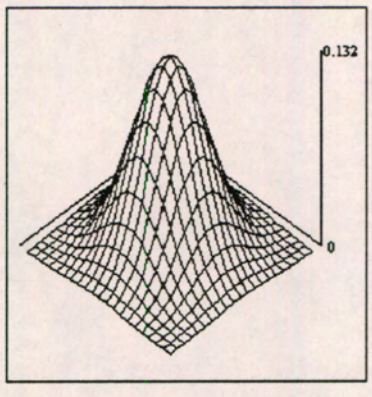


Figure 4.24: 2-D Gaussian Function

The first derivative in any direction with normal vector **n** can then be found by the partial derivative of G(x,y) in the direction of **n** i.e.

$$G_n = \frac{\partial G}{\partial n} = \mathbf{n} \cdot \nabla G$$

where the *del* operator ∇ indicates the gradient of $G(x,y)$ i.e.

$$\nabla G = \frac{\partial G}{\partial x}\hat{i} + \frac{\partial G}{\partial y}\hat{j}$$

The convolution of G_n with the image I is performed in each of the 4 directions as indicated in figure 4.13. Taking for example G_n in the x -direction we get:

$$\partial_x(I \otimes G(x,y)) = (I \otimes \partial_x G(x)) \otimes G(y)$$

Ignoring the constant term we have now got a solution using only the one-dimensional Gaussian functions G(x) and G(y) for the convolution.

$$H_x(x,y) = (I \otimes (-xe^{\frac{-x^2}{2\sigma^2}})) \otimes e^{\frac{-y^2}{2\sigma^2}} \tag{4.8}$$

Due to the symmetry of the two-dimensional Gaussian function assuming the σ is the same in the x and y directions, the convolution in the y-direction is found doing a similar computation.

The directions (1) and (2) in figure 4.13 correspond to the x - and y -directions respectively. The directions (3) and (4) correspond to a 45 degree rotation of the axes and the symmetry of (1) and (2) does not hold for example between (1) and (3) . G(x) and G(y) will thus have to be appropriately scaled to take into account this shift in axis.

If we observe the "distance" in the image over which the Canny function is spread we can calculate the amount of scaling that takes place in directions (3) and (4) corresponding to g_{xy} and g_{yx} of figure 4.19. The rotation of $\frac{\pi}{4}$ means that a function of width w in the x and y directions will have width $\sqrt{2}w$ in the directions of g_{xy} and g_{yx}, which effectively scales σ by the same amount i.e. $\sigma_{xy} = \sqrt{2}\sigma_x$. In this thesis project the rescaling was not done as only the pixel level edge position needed to be found by the Canny edge filter. Although not mathematically perfect experimentation showed that the final edge position is not affected.

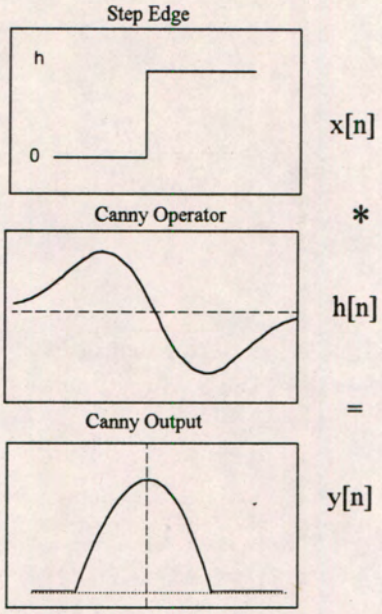


Figure 4.25: Convolution of the 1-D Canny operator with a step edge

Refer to figure 4.25 for an example of the step edge $x[n]$ being convolved with the Canny edge operator with response $h[n]$. The convolution output is indicated as $h[n] \otimes x[n]$.

Laplacian of Gaussian Edge Operator

The Laplacian of Gaussian (LoG) operator is directly related to the Canny Edge Detector. Where the Canny Edge Detector is formed by taking the first-derivative of the Gaussian smoothing function, the LoG operator is formed by taking the second derivative, or Laplacian of the Gaussian smoothing function [45] [56]. The results of the convolution between the input greyscale image and the edge-operators are stored in a 2-D array referred to as the "edge-enhanced image". For the Canny Edge Detector, the final edge position is found at the local maxima points of the edge-enhanced image. For the LoG operator, the final edge position is found at the zero crossings of the edge-enhanced image. Considering the fact that local maxima occur at points where the second derivative becomes zero, it is clear that both the Canny and LoG operators are based on the same mathematical principle.

The one-dimensional Gaussian function is given by:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} exp \frac{-x^2}{2\sigma^2} \tag{4.9}$$

and the second derivative :

$$G''(x) = \frac{x^2 - \sigma^2}{\sigma^4} G(x) \tag{4.10}$$

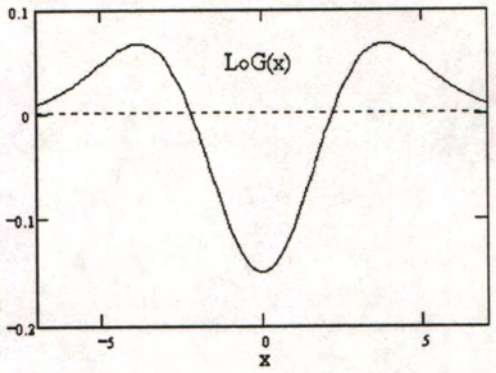
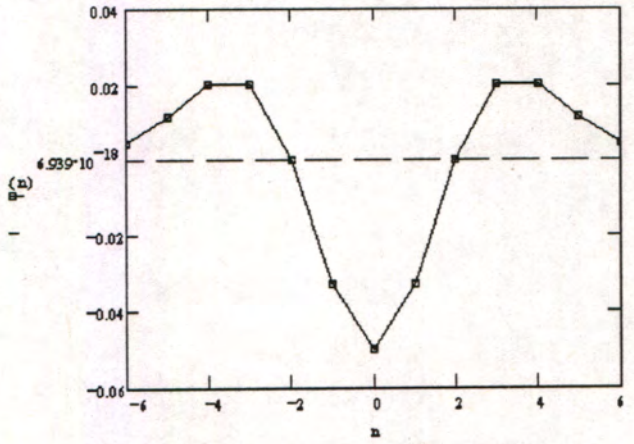


Figure 4.26: 1-D LoG Operator

Similar to the Canny edge operator the LoG operator has to be changed to a discrete convolution kernel by sampling the continuous function $G''(x)$ at integer locations of x. Figure 4.27 shows an example of a discrete LoG function with $\sigma = 2$ and width = 13. The resulting kernel $L[n]$ is shown in table form.



L(n)
0.004
0.012
0.02
0.02
0
-0.033
-0.05
-0.033
0
0.02
0.02
0.012
0.004

Figure 4.27: Discrete LoG function with the resulting convolution kernel $L[n]$

The frequency domain representation of the LoG function has the same characteristics as that of the Canny function. Theoretically if a continuous function $f(t)$ has frequency representation $F(\omega)$ then the first derivative $\frac{df(t)}{dt}$ has the frequency representation $j\omega F(\omega)$. The Fourier Transform for a LoG function with $\sigma = 2.2$ and width = 13 pixels is shown in figure 4.28. As with the Canny edge operator, the width of the LoG function depends on the amount of smoothing σ. If the width between the zero crossings of the LoG function is w where $w = 2\sqrt{2}\sigma$ then a kernel width of $4w$ is generally accepted [16] [94] and [37]. Note that this implies a larger kernel for the LoG function than the Canny function to get a similar frequency response. If we compare the Fourier transforms of the Canny function shown in figure 4.22 with that of the LoG function shown in figure 4.28 a hint to this conclusion is found. The peaks of the passband in the LoG response occur at a slightly higher frequency than that of the Canny function, and the passband is slightly wider. If the kernel were made a bit larger the peak frequency would decrease and the passband get narrower.

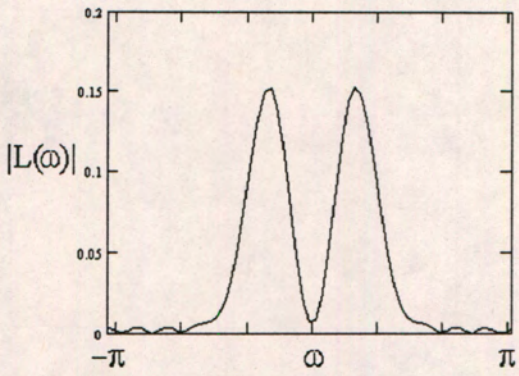


Figure 4.28: Frequency Response of the LoG edge operator with $\sigma = 2.2$ and width = 13

This function can also be extended to 2-dimensions as follows:

$$\nabla^2 G(x,y) = \frac{1}{2\pi\sigma^4}\left[2 - \left(\frac{x^2+y^2}{\sigma^2}\right)\right] exp\frac{-x^2+y^2}{2\sigma^2} \tag{4.11}$$

where ∇^2 is the Laplacian operator indicating the second derivative such that

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2}\hat{i} + \frac{\partial^2 G}{\partial y^2}\hat{j}$$

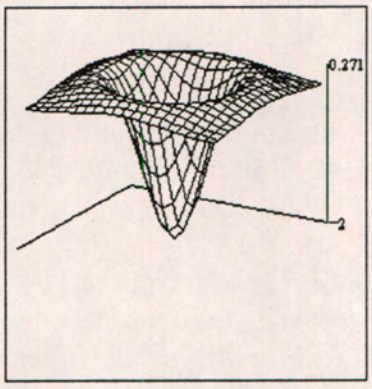


Figure 4.29: 2-D LoG Operator

Refer to figure 4.30 for an example of the step edge $x[n]$ being convolved with the Laplacian of Gaussian edge operator with response $h[n]$. The convolution output is indicated as $h[n] \otimes x[n]$.

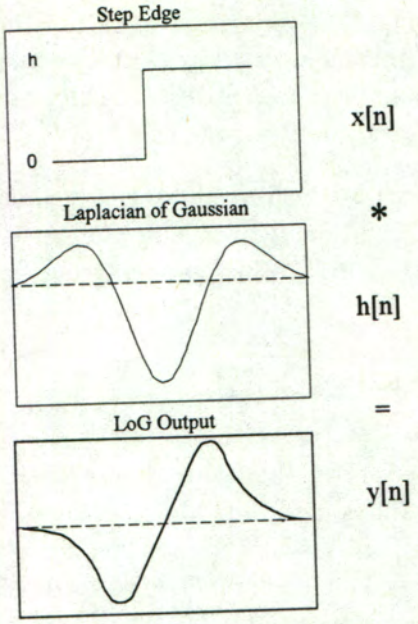


Figure 4.30: Convolution of the 1-D LoG function with a step edge

The final edge positions are found by looking for zero crossings in the edge enhanced map, which stores the result of the convolution between the image and the Laplacian of Gaussian mask. Figure 4.31 shows the resultant binary edge map for an aerial image convolved with an 11x11 LoG mask with $\sigma = 1.4$

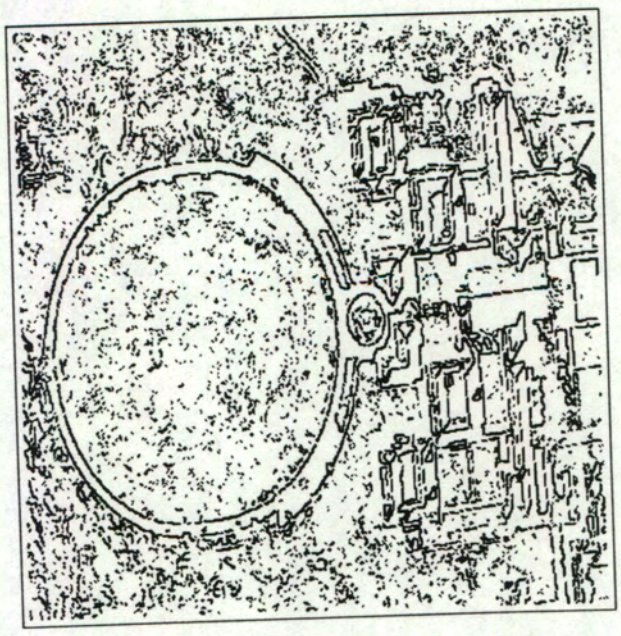


Figure 4.31: Binary edge map from convolution of LoG operator with $\sigma = 1.4$

The zero crossings are detected by comparing the central pixel to its eight neighbours in a 3x3 neighbourhood as follows: If the central pixel is less than the threshold $-g_t$ and one of its eight neighbours is greater than the threshold g_t, or if the central pixel is greater than the

threshold g_t and one of its eight neighbours is less than the threshold $-g_t$ then the pixel is marked as an edge point. The threshold value g_t relates to the slope of the line that has a zero crossing. A strong edge would be represented by a high slope, so increasing the threshold would suppress weak edges and mark only strong edges.

An advantage of the gradient- based operators above the LoG operator is the ease of location of the final edge positions. Computationally it is a lot more economical to apply thresholding followed by maxima detection than it is to find zero crossings in an edge-enhanced map.

Preservation of Moments Edge Detector

The sub-pixel edge detector proposed by Tabatabai and Mitchell [98] is different from the first- and second derivative based methods described above. Where edge detection was previously done by first performing edge enhancement then looking for local maxima (Local Gradient, Canny) or zero crossings (LoG) the final sub-pixel edge position is immediately found from a statistical analysis of the input greyscale data.

In digital images, a 1-D cross-section of an edge can be characterized as a set of greyvalues f_i that is either increasing or decreasing in value, depending on the direction of the edge. An *ideal* step edge on the other hand is a sequence of greylevels at one level h_1 followed by a sequence of greyvalues at a different level h_2. The edge operator proposed generates an ideal edge from the input data by locating the ideal step edge position at which the first three moments of the input data sequence are preserved. The first three moments are defined as

$$\bar{m}_i = \frac{1}{n}\sum_{j=1}^{n} f_j^i \quad \text{for i = 1,2,3} \tag{4.12}$$

where f_j is the greyvalue at point j and n is the size of the data sequence.

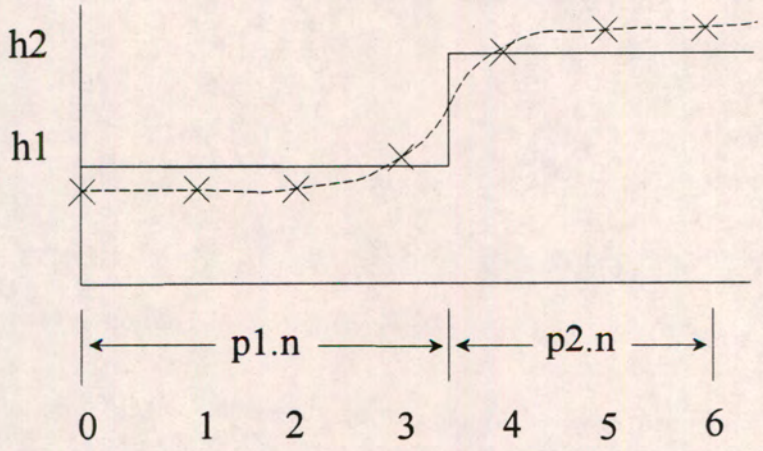


Figure 4.32: Sample Data and the Ideal Step Edge with the same first three moments

By modelling the ideal step edge to have the same moments as the data, we can solve for the ideal edge position at position k in the input sequence. The fractions p_1 and p_2 are the fractions of the sample length n at which the ideal step edge has greyvalues of h_1 and h_2 respectively. For example if the input data has width $n = 7$ and p_1 evaluates to 0.4 then the position k at which the ideal step edge is placed is at $k = p_1 n = 2.8$. The unknowns p_1, p_2, h_1, h_2 and k are solved for by solving the three equations

$$\sum_{j=1}^{2} p_j h_j^i = \bar{m}_i \quad \text{for i = 1,2,3} \tag{4.13}$$

where

$$p1 = \frac{k}{n}$$

and

$$p_1 + p_2 = 1$$

to solve for the unknowns p_1, p_2, h_1 and h_2.

The solutions to these equations are given by Tabatabai and Mitchell [98] as

$$p_1 = \frac{1}{2}\left[1 + \bar{s}\sqrt{\frac{1}{4+\bar{s}^2}}\right] \tag{4.14}$$

$$p_2 = 1 - p_1 \tag{4.15}$$

$$h_1 = \bar{m}_1 - \sigma\sqrt{\frac{p_2}{p_1}} \tag{4.16}$$

$$h_2 = \bar{m}_1 + \sigma\sqrt{\frac{p_1}{p_2}} \tag{4.17}$$

where

$$\bar{s} = \frac{\bar{m}_3 + 2\bar{m}_1 - 3\bar{m}_1\bar{m}_2}{\sigma^3} \tag{4.18}$$

$$\sigma^2 = \bar{m}_2 - \bar{m}_1^2 \tag{4.19}$$

From the above results it should be noted that the edge location at $k = np_1$ may not be at an integer location, and the edge position has thus been found to a sub-pixel level without doing

any interpolation. Note that the direction of the edge has to be taken into account when the sub-pixel edge is found. In this example we have assumed that we are dealing with an upgoing edge i.e. a greylevel transition from low to high. The edge can also go from high to low, in which case $k = np_2$. The edge moments are the same, but instead of treating the data with the time variable j increasing from left-to-right we change the direction of our time-axis to increase from right-to-left.

Note that the edge location k is invariant under scaling and translation of the input data. Thus if

$$z_i = af_i + b \quad \text{for i = 1,2..n} \tag{4.20}$$

where a and b are constants, then using f_i and z_i as input will give the same edge location.

In this thesis project the sub-pixel edge position is found only for points on a feature boundary. Once features have been extracted all the discrete feature boundary points are refined to a sub-pixel boundary point. At each boundary point the edge direction θ_n is calculated as

$$\theta_n = tan^{-1}\frac{g_y}{g_x} \tag{4.21}$$

where g_y and g_x are the greyscale gradients in x and y respectively. This step would typically take place during the actual edge enhancement, with the edge direction stored in an array.

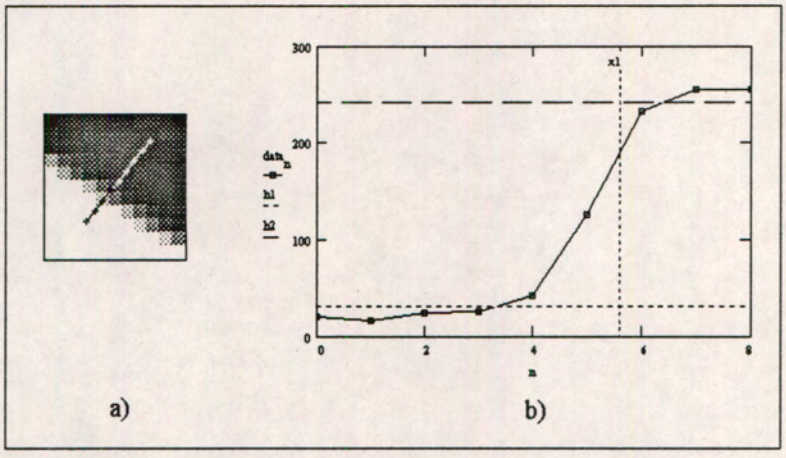


Figure 4.33: Resampled line along edge normal a) and Corresponding greyscale data and equivalent ideal step edge b)

Points along this direction are resampled on both sides of the boundary point, as shown in figure 4.33 a). In this case four points on either side of the boundary point were located to a sub-pixel level as

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} kcos\theta_n \\ ksin\theta_n \end{pmatrix} \tag{4.22}$$

where k is in the range $-4 \ldots 4$ and (x_n, y_n) is the centre of the original boundary point.

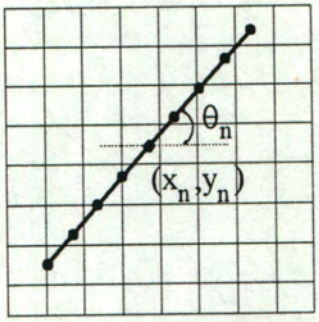


Figure 4.34: Calculation of sub-pixel points along the edge normal

These points are resampled using bilinear interpolation, which is discussed in more detail in the appendix. The resampled greyscale profile in the direction θ_n is shown in figure 4.33 b). The calculated heights h_1 and h_2 are shown and the final edge position is indicated by the dotted line. In this example the edge position k is found at 5.58, which is measured from position 0 on the line, indicated by the cross on the top right end of the line shown in 4.33 a). From this starting position it is simple to find the sub-pixel edge position from k and θ_n.

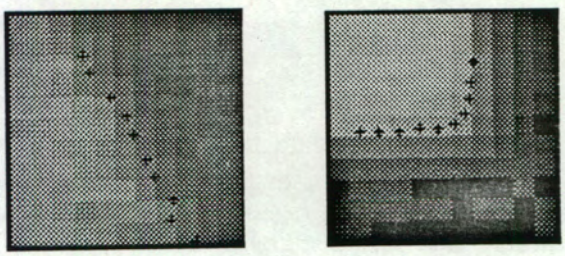


Figure 4.35: Sub-pixel edge positions for a straight line and a corner

In figure 4.35 the sub-pixel edge positions are shown for a straight line segment and a corner. The edge positions are indicated with a cross on the zoomed image data.

The method of moment based edge detection can be extended to multiple edge data to find for example a pair of edges, or to two dimensions using a line equation model in a circular disk [98] (see figure 4.36)

An ideal edge line can be fit to the 2-D surface at level h_1 to one side of the line and at level h_2 on the other side of line. The same method of the preservation of moments is now used to solve for the edge location in 2-D.

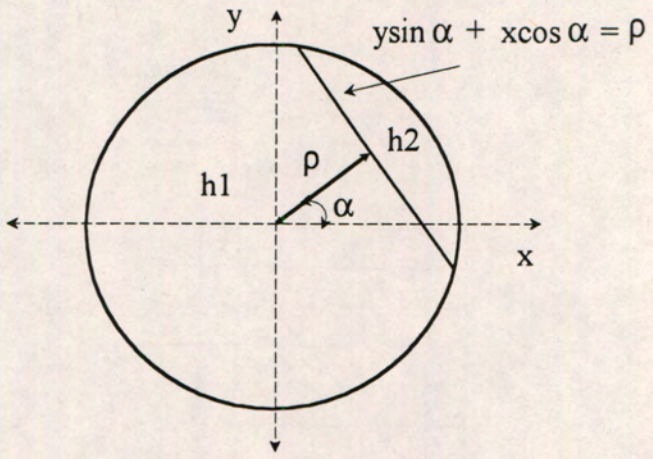


Figure 4.36: Ideal Edge line equation as a function of α and ρ

If we can denote an ideal edge element by $U(x, y, sin\alpha, cos\alpha, r, h_1, h_2)$ where α is the angle normal to the line and r is the radial distance to the line, then we can model the ideal step edge as

$$U(x, y, sin\alpha, cos\alpha, \rho, h_1, h_2) = h_1 \quad \text{if } ysin\alpha + xcos\alpha \leq \rho \tag{4.23}$$

$$U(x, y, sin\alpha, cos\alpha, \rho, h_1, h_2) = h_2 \quad \text{if } ysin\alpha + xcos\alpha > \rho \tag{4.24}$$

and solve for all the parameters of the line by equating the moments of the data to the moments for a line describing an ideal step-edge .

In this thesis project the moment-based edge detector was not extended to two dimensions due to the increase in computation that is needed.

4.1.3 Edge Strengthening

Most linear edge detectors have a disadvantage in not using the edge information already available in the vicinity of the possible edgel. Edge strengthening uses the edge strengths surrounding an element to either strengthen or weaken an edge.

The reader is referred to figure 4.37 for an example of the edge enhanced map for a PCB image with a Canny filter with $\sigma = 1$.

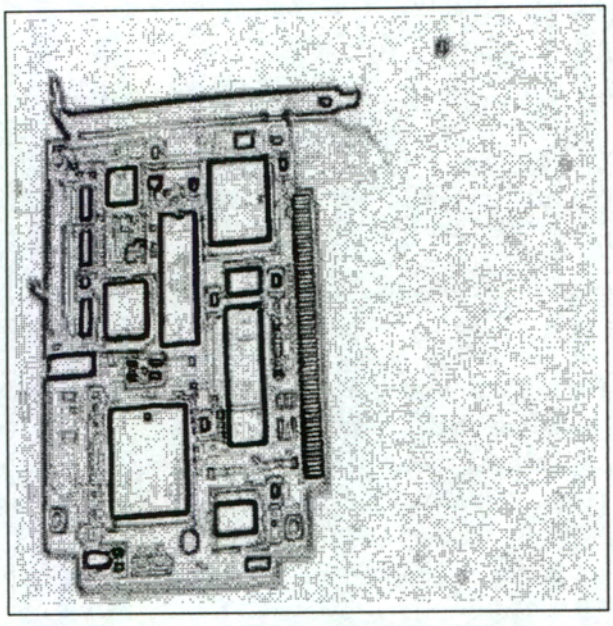


Figure 4.37: Edge enhanced map for a PCB image with a Canny filter with $\sigma = 1$.

The edge strengthening algorithm used forms an Orientation Response Histogram(ORH) of the edge strengths in the four primary edge-normal directions shown in figure 4.38 . The ORH represents the sum of the current, the previous and the next pixel edge strengths in each of the four directions [35].

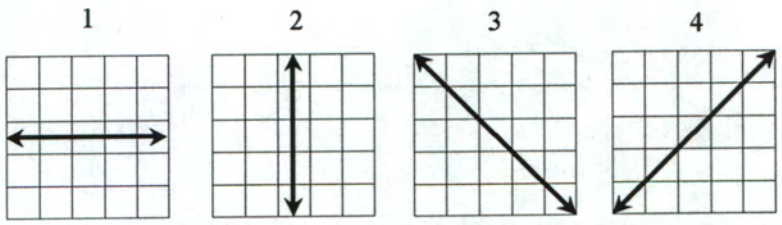


Figure 4.38: Major edge directions in a 5x5 neighbourhood

The edge element is then replaced by the maximum of the four ORH values, with the direction being the direction in which this maximum has occurred. This edge strengthening has the advantage that a weak link in an otherwise strong edge chain is compensated for.

As an example, we compute the ORH for a 3x3 neighbourhood with a prominent edge in direction (4), thus giving a maximum response in the normal direction (3). The edge strengths obtained from the edge enhanced map in the 3x3 neighbourhood is shown below.

$$\begin{pmatrix} 100 & 70 & 30 \\ 50 & 50 & 50 \\ 30 & 70 & 100 \end{pmatrix}$$

Applying the ORH algorithm to this data we get:

orh[1]= 50 + 50 + 50 = 150
orh[2]= 70 + 50 + 70 = 190
orh[3]= 100 + 50 + 100 = 250
orh[4]= 30 + 50 + 30 = 110

The maximum value of these is 250 , which occurs in direction (3). The centre pixel's edge strength is thus replaced by this value, and the tangential edge direction recorded as the normal to direction (3) i.e.direction (4).

We see from this example that the pixels with edge strength 100 on either side of the central pixel for direction (3) have effectively enhanced the weak link of 50 in the edge chain.

This method can be extended to strengthen corners and line junctions. In both these cases the ORH will have 2 maxima that are relatively close to each other, as at for example a corner there will be 2 prominent edges close to the corner . The corner or junction can thus be enhanced by taking the two local maxima ORH values into account and replacing the centre pixel edge strength by an average of these two. The scope of the ORH can also be increased by observing a larger neighbourhood i.e. 5x5 or 7x7 pixels to search for corners and line crossings.

Not taking two directions into account could mean that a strong corner or junction edge element can be effectively weakened using this algorithm. Experimentation has shown that gaps do appear at corners and junctions due to using linear edge operators on a 2-D image , and these will have to be compensated for by a process other than edge strengthening. In this thesis project gaps appearing in edge chains are bridged using the line following technique that will be described in detail later in this chapter.

4.1.4 Maxima Detection

In order to obtain the final edge position the local maxima must be found in the edge-enhanced map. As an example, we take a 1-D cross section from the edge enhanced map, shown in figure 4.39. Figure 4.39 a) shows the progression of a 128x128 sub-image from greyscale intensity image at the top, edge enhanced map in the middle to the binary edge sub-image at the bottom. The horizontal line in the middle of each sub-image indicates where the 1-D cross section is taken. The greyscale intensity profile is shown in the top of 4.39 b), followed by the cross-sections through the edge enhanced maps resulting from the gradient-sum method with $n = 4$, Canny with $\sigma = 2.2$ and Canny with $\sigma = 0.7$ respectively.

The local maxima points corresponding to the final edge position are found by thresholding the data then searching for local maxima. All the local maxima points in the edge enhanced image represent edges to which the edge filter used is sensitive. To extract only prominent edges and ignore weak, noisy edges thresholding is applied. The height of each local maxima point is proportional to the edge strength and local maxima at a height lower than the set threshold can be discarded. The resultant image with only the final edge positions marked is referred to as the *binary edge map.*

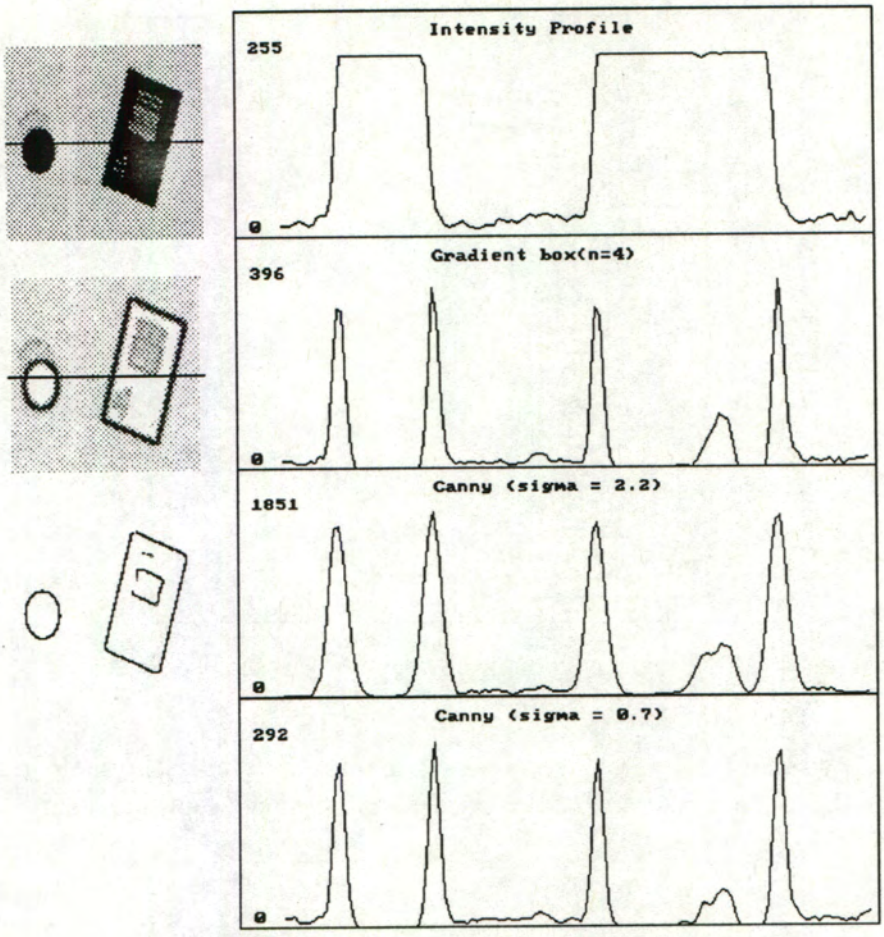


Figure 4.39: 1-D cross-section of edge detection results

The threshold can be chosen in a local or a global sense. A *global* threshold is a single threshold that is applied all over the image, where a *local* threshold is a threshold that is applied to only a part of the image. Local thresholding is applied to windows that form subsets of the whole image. Breaking up an image into smaller image windows is referred to as *tesselation*. Different methods of thresholding the gradient edge map such as adaptive thresholding discussed by Jeong and Kim [60] and Nana [78] can be found in the literature. Bergholm [10] and Lacroix [67] report that the emphasis of the thresholding technique should be on keeping significant edges intact while insignificant edges such as shadows and noise are suppressed.

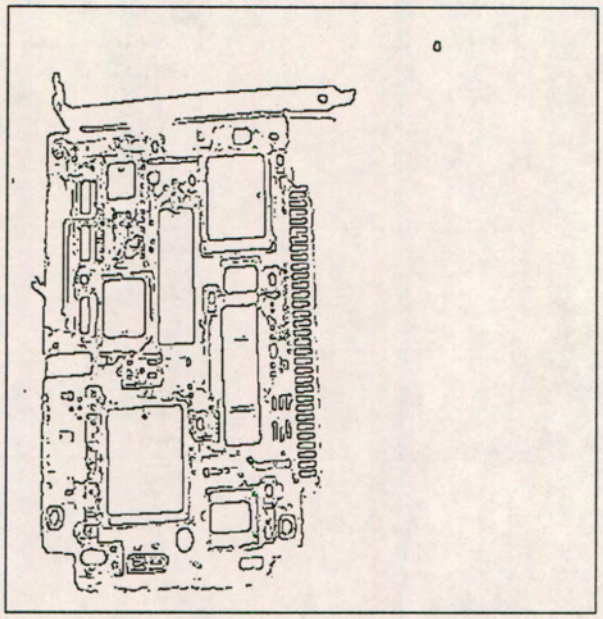


Figure 4.40: Binary edge map from global threshold $g_t = 0.7g_{max}$

Figure 4.40 shows the resultant binary edge map with the global threshold g_t at $0.7g_{max}$ where g_{max} is the global maximum of the edge enhanced image shown in figure 4.37.

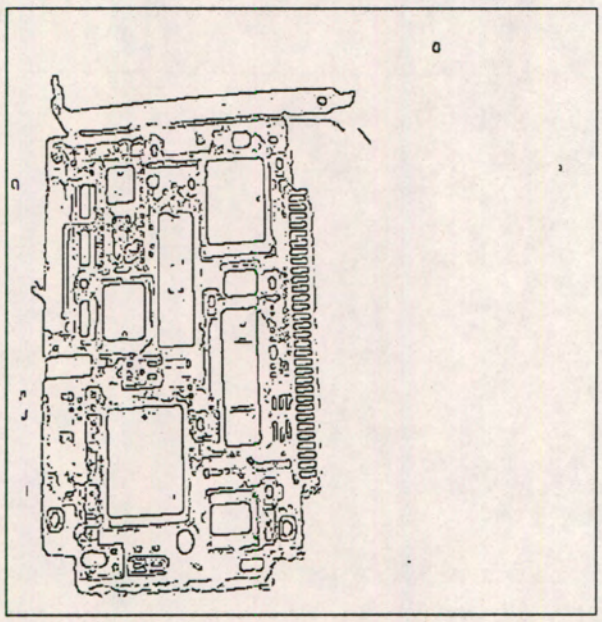


Figure 4.41: Binary edge map from global threshold $g_t = g_{ave} + 3\sigma_g$

Figure 4.41 shows the resultant binary edge map with the global threshold g_t at $g_{ave} + 3\sigma_g$ where g_{ave} is the global average of the edge enhanced image and σ_g is the standard deviation. Note that a similar amount of detail has been captured by both the global thresholds. The amount of detail can be controlled by the threshold. If more detail is wanted the threshold

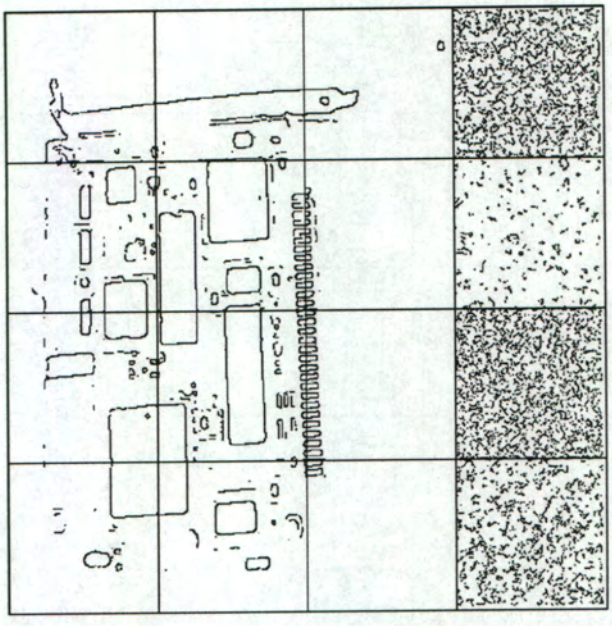


Figure 4.42: Tesselated binary edge map from local threshold $g_n = 0.4\bar{g}_{max}$

is decreased and if less detail is wanted i.e only strong edges found then the threshold is increased. Thresholding can be made semi-automatic by allowing the user to manually increase or decrease the threshold. In this thesis project automatic thresholding was used with $g_t = g_{ave} + 3\sigma$ as shown in figure 4.41. Although a lot of unwanted detail is still present it can be discarded at a later stage by the feature extraction algorithm.

In comparison we look at the binary edge map obtained by tesselating the image into 16 sub-images of 128x128 pixels each. Figure 4.42 shows the resultant tesselated binary edge map with the local threshold g_n for each sub-image at $0.4\bar{g}_{max}$ where $\bar{g}_{max}$ is the local maximum for sub-image n. Note that even at a much lower fraction of the local maximum point there is still less detail visible than at a higher fraction of the global maximum. The four sub-images on the right show some detail which is not visible in figure 4.41. These entire sub-images consist of weak edges and are not of interest for image matching and can be discarded.

In figure 4.43 we can compare the tesselated binary edge map with the other binary edge maps. The local threshold g_n for this image is set at $\bar{g}_{ave} + \sigma_n$, where $\bar{g}_{ave}$ is the local average of the sub-image n and σ_n is the standard deviation for the sub-image n. Note that a similar amount of detail is visible than in figure 4.42 , but again there is less detail at a relatively lower threshold than in figure 4.41 where the threshold is set globally.

In figure 4.44 the edge enhanced map for a circular feature can be visualized as a three dimensional surface. Instead of a one-dimensional search in four directions for local maxima as is done in this thesis project, local maxima can be found by searching for local maxima in a 3x3 neighbourhood around a point. This two-dimensional search is similar to the search for zero crossings using the LoG filter.

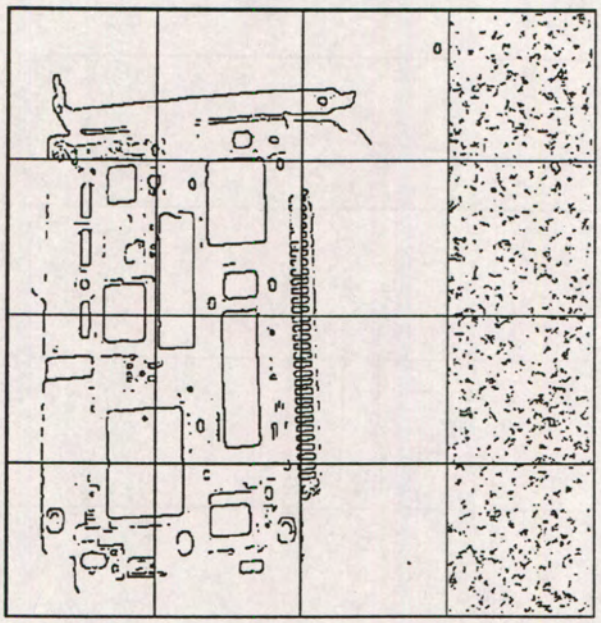
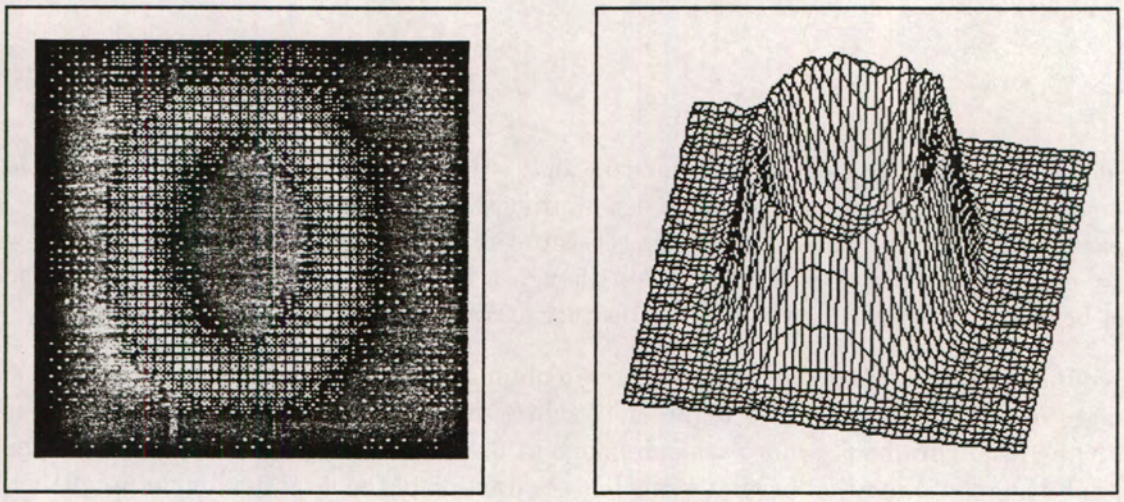


Figure 4.43: Tesselated binary edge map from local threshold $g_n = \bar{g}_{ave} + \sigma_n$



E E

Figure 4.44: 3-D visualization of edge enhancement results

4.2 Line Following

The binary edge map obtained from edge enhancement followed by edge strengthening and maxima detection gives an indication of where the edges in the image are but does not give any indication of which edge pixels, referred to as edgels, are grouped together to form an edge chain. Linking together and storing the (x, y) coordinates of the line features will be referred to as *vectorizing*.

In this project a line following algorithm is used that attempts to link an edgel to its neighbours in a 3x3 neighbourhood. The algorithm attempts to keep consistency in the direction of the edge chain by recording the direction in which a previous edgel has been linked and using this information to decide in which direction the search for the next edgel in the edge chain must start.

4.2.1 The Circular Search Algorithm

The Circular Search algorithm tracks around the outside of line features in a clockwise direction [86]. The directions from the centre pixel to the eight neighbouring pixels in a 3x3 neighbourhood are labelled in a clockwise direction i.e.

$$\begin{matrix} 7 & 0 & 1 \\ 6 & * & 2 \\ 5 & 4 & 3 \end{matrix}$$

The central pixel, marked as $*$, represents the latest edgel found in the edge chain.

The algorithm starts in the top,left corner of the binary edge map (position (0,0)) and searches from top to bottom and left to right until an edge pixel is found. From here the circular search algorithm is applied sequentially as follows, starting the first search for a neighbouring edgel in direction 0:

- Test the starting pixel position for the presence of an edgel
- If an edgel is not found at the starting position then test the neighbours of the current pixel in a clockwise direction by incrementing the direction of the search for an edgel until an edgel is found
- When an edgel is found it becomes the current edgel and the starting search position for the next link in the edge chain is determined by the direction in which the edgel was found, as indicated in figure 4.45
- Iterate the search until a closed line feature has been completely tracked or the end of a line is indicated when no more links to the edge chain can be found

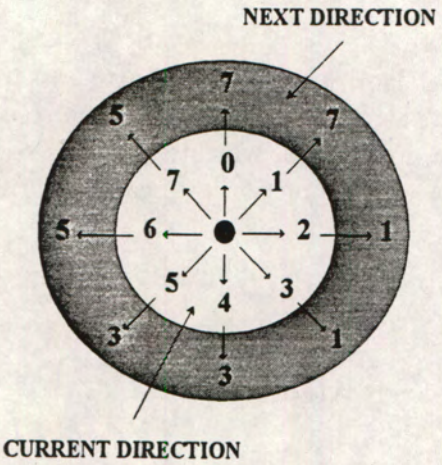


Figure 4.45: Circular search grid for line following

Figure 4.45 shows the circular rotation ring which indicates how the starting position for the next search is determined by the direction in which the current edgel was found. For example if the current edgel was found in position (1) then the search for the next edgel will start in direction (7).

To give an example of how the circular search algorithm works the algorithm will be compared to a similar algorithm. This algorithm does not adapt the starting position for the next search, but always starts searching for a neighbouring pixel at the same position (0). The circular searching algorithm adapts the starting position for each edgel in the edge chain according to the rotation ring. The different results obtained by these two methods can conveniently be highlighted using the example shown in figure 4.46

col →

row ↓	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
2	0	0	1	1	1	0	0
3	0	1	1	1	1	1	0
4	0	0	1	1	1	0	0
5	0	0	1	1	0	0	0
6	0	0	0	0	0	0	0

Figure 4.46: Binary Image Feature

In the image array shown, edge pixels are labelled as 1 and the background as 0. The algorithms first check all pixels from left to right and top to bottom until the first edgel is found at position (1,3) in the array. From here the circular search algorithms start looking for edgels linked to the edgel at position (1,3). The 3x3 neighbourhood centered at (1,3) looks as follows:

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{matrix}$$

The search for the next edgel starts at position (0) and is incremented until the next edgel is found in direction (3) at position (2,4). This position now becomes the centre of the current 3x3 neighbourhood i.e.

$$\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{matrix}$$

From the rotation ring of figure 4.45, we see that if the current edgel was found in direction (3) then the search for the next edgel must start in direction (1). With this starting direction, the circular search iterates until the next edgel is found at position (3,5) in direction (3).

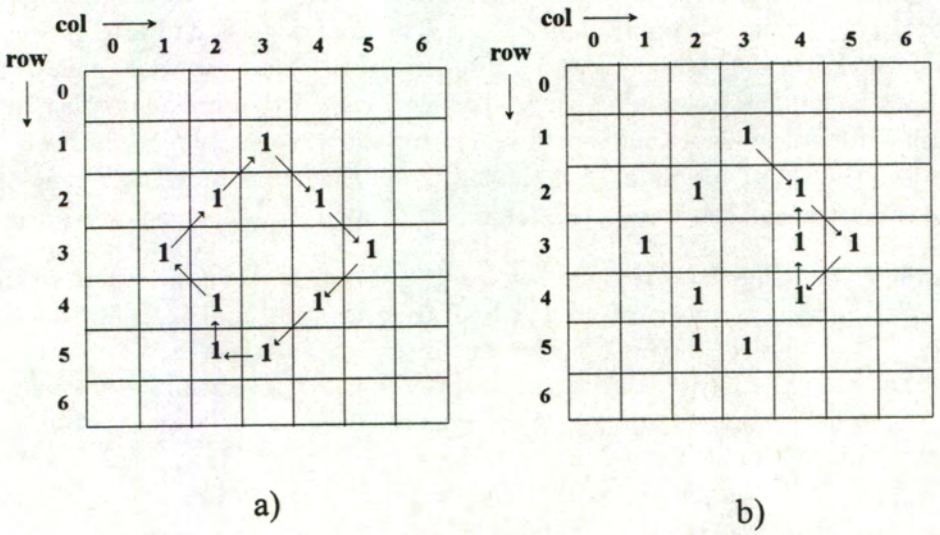


Figure 4.47: Line following results from adaptive starting position a) and constant starting position b)

This procedure is repeated until the search algorithm has tracked all the way around the feature outline, as shown in figure 4.47 a), where the feature boundary points found by the line-following algorithm are indicated by the number 1. The directions from each boundary edgel to the next is indicated by an arrow. The circular search algorithm stops as the last edgel found is the same as the very first edgel i.e. the entire polygon has been tracked.

If the feature outline shown were to be chain coded, the chain code for this polygon feature would be 335560711. See section 5.4.1 for a more detailed description of the Freeman Chain Code.

Figure 4.47 b) shows the result obtained by the circular line following algorithm if the starting position for the new search is not updated but always kept at direction (0). The first four boundary edgels are exactly the same as in figure 4.47 a). With the edgel at (4,4) in the array as the current element, the 3x3 neighbourhood looks at follows:

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{matrix}$$

The next rotation ring would indicate that the next search should start in direction (3), which would lead to the correct boundary pixel being found at position (5,3). If however the next search direction is not updated and the next search begins in direction (0) then the (incorrect) "boundary pixel" at position (3,4) will be found. From this position the next "boundary pixel" will be found in direction (0) at position (2,4), causing an infinite loop as shown in figure 4.47 b). This example shows why it is necessary to use the direction information to decide where to look for the next link in the edge chain.

In theory only boundary pixels of the feature will be marked as edgels and the interior pixels will be marked as background (0). If this were the case then always starting the search in direction (0) will still cause an infinite loop. The next edgel found after position (4,4) will be (3,5), which once again causes an infinite loop. In this case a record must be kept of the number of times that a boundary element has been visited. If a boundary element has already been visited then the algorithm should search for another possible link in the edge chain. In this project the (x, y) coordinates of the feature outlines are stored and then the boundary element is deleted to ensure that the line following algorithm does not trace back onto itself.

An example of when a feature boundary can be traced back onto itself is when the feature boundary is more than one pixel in width. During the final maxima detection stage of edge detection, local maxima in the 2-D edge enhanced map are searched for using a linear one-dimensional search, which could lead to feature boundaries that are more than one pixel in width. Some line thinning techniques as proposed by Rosenfeld amongst others have been investigated but not implemented in this thesis project [84] [23] [3]. These thinning algorithms all try to find the "skeleton" of a broad line feature without breaking any of the connections between the important "skeleton" pixels.

4.2.2 Bridging Gaps in Edge Chains

Due to noise in images, occlusions, shadows and thresholding amongst others the binary edge map obtained from edge enhancement and maxima detection will inevitably have gaps in the edge chains. The edge strengthening method described has been shown through experimentation to improve matters by strengthening or weakening a possible edgel according to the edge strengths and directions of the neighbouring pixels. A problem with gaps does still exist though, especially at corner points.

In this thesis project the circular searching algorithm has been modified to attempt to fill in gaps in edge chains. The modifications to the algorithm consist of creating a hypothesis

about the location of the "missing link" in the edge chain and then checking the validity of the hypothesis.

The edge linking algorithm can be explained through the examples shown in figure 4.48

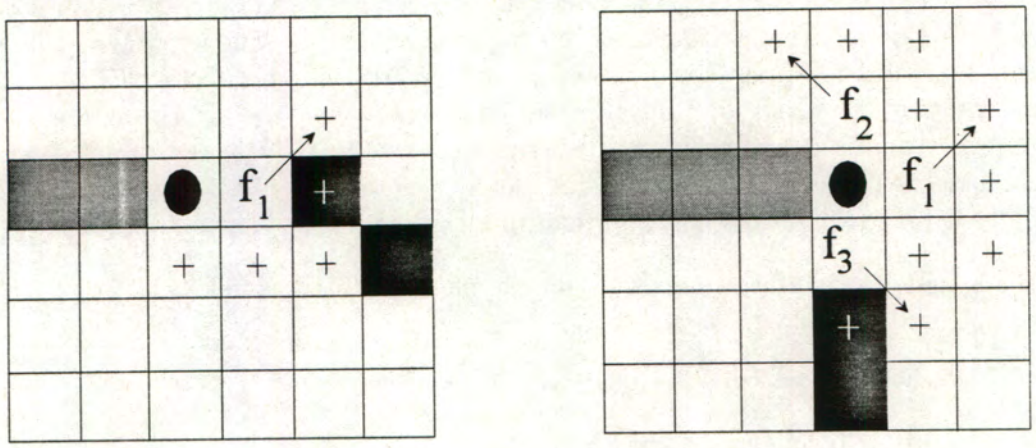


Figure 4.48: Bridging a gap in a straight line a) and at a corner b)

Figure 4.48 a) shows a gap in a straight line segment. Assume that the circular searching algorithm tracks the line in a clockwise direction. The edgels already found are indicated by the hashed blocks. With the current edgel, indicated by the circle, in the centre of the neighbourhood, it is clear that with the edgels that have already been found deleted that no link will be found in the 3x3 neighbourhood around the current edgel. This would indicate either a node i.e. the endpoint of an ARC or that there is a gap in the edge chain.

An assumption as to where the gap occurs is made by taking into account the direction in which the current edgel was found. In this example the current edgel was found in direction (2), and the hypothesis is made that the gap occurs in direction (2) relative to the current edgel, marked f_1. This hypothesis must now be tested by examining the 3x3 neighbourhood around pixel f_1. The rotation ring of figure 4.45 shows that the starting position for the search is direction (1), indicated by the arrow. Only this pixel plus the next four pixels in the neighbourhood, marked with crosses, are checked for an edgel. Only five directions are checked as a rotation of more than four elements constitutes a change of more than 180 degrees in direction in a 3x3 neighbourhood, which is not acceptable. The assumption is made that a piecewise smooth curve cannot change direction by more than 180 degrees in a 3x3 neighbourhood. In this example the connecting edgel is found after the first iteration in direction (2) and the hypothesis that the pixel labelled f_1 is the missing link in the edge chain is verified.

The example shown in figure 4.48 b) shows the case of a gap occurring at a corner of a feature, which is a situation that frequently occurs. As in the previous example the first hypothesis is that the gap occurs in the same direction as the current edgel, indicated by the circle. The first possible gap position is indicated as f_1 in direction (2) from the current edgel. Starting

in direction (1) as indicated by the arrow, the following five pixels in the 3x3 neighbourhood centered at f_1 are inspected. In this example no linking pixels are found and a new hypothesis must be formulated. From here one of two assumptions are tested - the gap occurs at ninety degrees in either a clockwise or an anti-clockwise direction relative to the first assumption f_1. The first assumption in this case was that f_1 is in direction (2), so the next assumption would be that the gap appears in either direction (0) or direction (4). The hypothesis that the gap appears at position (0), marked f_2 is tested first. Starting from direction (7) shown by the arrow, the next five pixels are inspected and no link is found. The algorithm now checks the hypothesis that the gap appears in direction (4) shown as f_3. Starting in direction (3) the linking pixel is found in direction (4) and the hypothesis is verified. The gap marked f_3 is thus added to the edge chain and the algorithm continues from this linking pixel.

In pseudo-code the gap-bridging algorithm can be represented as follows:

```
if( no linking pixel found ) /* indicates either a node or a gap */
      {
      if(no Hypothesis checked )
            {
            current element = f1;
            direction = current direction;
            Check Neighbourhood(current element,direction); /* checks for linking pixels */
            if( Link Found )
               out of loop;
            else
               {
               Hypothesis 1 checked;
               out of loop;
               }
            }

      if( Hypothesis 1 checked )
            {
            current element = f2;
            direction = [current direction-2]mod8;
            Check Neighbourhood(current element,direction); /* checks for linking pixels */
            if( Link Found )
               out of loop;
            else
               {
               Hypothesis 2 checked;
               out of loop;
               }
            }

      if( Hypothesis 2 checked )
            {
            current element = f3;
            direction = [current direction+2]mod8;
            Check Neighbourhood(current element,direction); /* checks for linking pixels */
            if( Link Found )
               out of loop;
```

```
        else
            {
            Hypothesis 3 checked;
            out of loop;
            }
        }

    if( Hypothesis 3 checked )
        {
        current = NODE;
        out of loop;
        }
}
```

The above algorithm has successfully been used to bridge single gaps appearing in the edge chain. The scope of the gap-filling can be increased by increasing the size of a neighbourhood when a hypothesis is checked. The same algorithm can thus be slightly modified to bridge gaps of more than one pixel in the edge chain. This however could lead to line segments that don't belong together being linked together, indicating that the maximum size gap that can be bridged in an edge chain shouldn't be more than 3-4 pixels, depending on the expected minimum distance between significant features in the image.

4.2.3 Following Open Line Features

Open line segments, defined as ARCs, are indicated when the line following algorithm finds a *node*, which is an edgel that cannot be linked to any other edgel in close proximity. As the algorithm for bridging gaps in edge chains indicates, a node is found only when all three hypotheses have failed. When a node is found it is stored as one of the ends of the ARC or line segment and the search position is changed to try and find the other node. See figure 4.49 for an example of vectorizing an ARC.

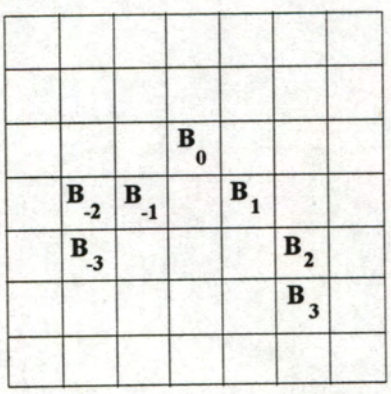


Figure 4.49: Vectorizing an open line segment (ARC)

The line following algorithm does the top-down and left-to-right search of the binary edge map with line edgels marked $B_{-3}..B_3$ as shown. The coordinates of the starting edgel B_0 is

stored and the search continues in the clockwise direction around the outside of the arc until a node is found at B_3. The feature vector which holds the coordinates of the line edgels is thus ordered as $F_n = B_0, B_1, B_2, B_3$. The coordinates of the node B_3 are also stored separately. The line following algorithm has effectively reached a "dead-end", and the current position has to be updated so that the rest of the curve can be traced.

The search window is now moved back to the starting position at B_0. The edgels $B_0..B_3$ have been deleted, so the next edgel found will be B_{-1}. To keep continuity in the order of the edge chain the feature vector must be swapped around, making the node B_3 the starting point of the edge chain. The feature vector will now be ordered $F_n = B_3, B_2, B_1, B_0, B_{-1}$. Note that the clockwise search direction now means that the arc is tracked from the inside, finding B_{-3} before finding B_{-2}. This can be seen as a flaw in the circular searching algorithm. A better approach would be to change the search direction to anti-clockwise when the search for the second part of the ARC is done. This could be done by simply decreasing the search direction instead of increasing it. With the search direction changed due to the first node being found, the correct feature vector will thus be ordered $F_n = B_3, B_2, B_1, B_0, B_{-1}, B_{-2}, B_{-3}$ with the two nodes B_3 and B_{-3} of the ARC at the endpoints of the edge chain as expected.

4.3 Feature Editing

As described in section 4.2.2, the line following algorithm can be hampered by gaps in edge features or by line features that are in close proximity. The more complex a scene becomes the greater is the chance that some line features will be incorrectly vectorized. A limited amount of editing has been allowed for either after the binary edge map has been found or after feature tracking. If editing takes place after the features have been tracked, the features have to be tracked again to enforce the changes made during editing. If the binary edge map is edited only pixel editing steps are allowed.

Feature editing introduces the only step in the entire matching process that needs user input. The user can edit the feature vectors graphically using the mouse, and the following editing steps are available:

- Arc Deletion
- Feature Deletion
- Arc Linking
- Line Addition
- Pixel Add/Delete

Arc Deletion allows the user to indicate the begin and end points of a sub-arc on a line feature that must be deleted.

Feature Deletion allows the user to delete an entire ARC or POLYGON feature, which is indicated using the mouse.

Line features can be merged using the *Arc Linking* option. This option is only available for feature defined as ARCs. The user indicates a node on each of the ARC features with the mouse. The features are then merged by connecting the two nodes with a straight line.

A straight line can be added to a feature using the *Line Addition* option. The user indicates the begin and end points of the line which is to be added to the feature.

The editing options that are used most often are the *Pixel Add/Delete* options. With these options individual pixels on any feature boundary can be removed or added.

Before any of the editing steps are effected the user is asked to confirm the editing step. If a mistake is made the mistake can thus be rectified or that specific editing step can be cancelled. After all the editing has been done on the feature image the features are re-vectorized, which updates any changes in the feature boundaries.

Figure 4.50 shows three examples of polygon features that have been incorrectly vectorized. From top to bottom the figure shows the original greyscale sub-images, the binary edge maps and the vectorized feature boundaries.

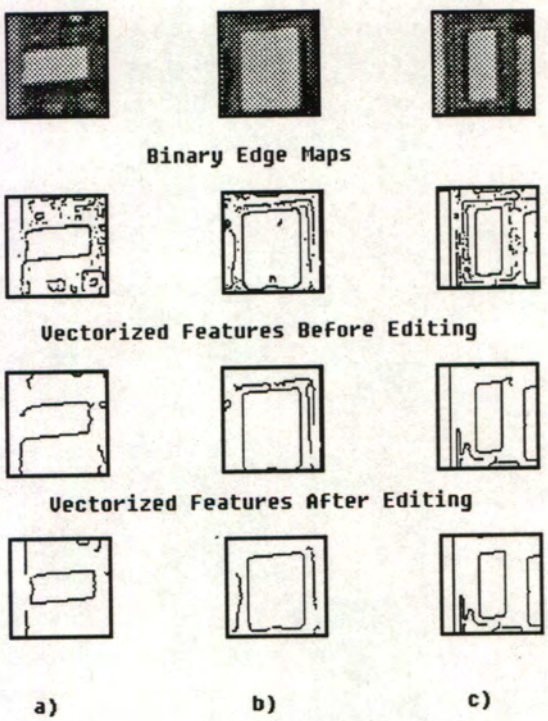


Figure 4.50: Incorrectly Vectorized Polygon Features

Figure 4.50 a) shows an example of a prominent edge that coincides with the boundary of a rectangular feature due to the viewing angle. At the bottom-left hand corner of the rectangle three lines meet at the corner point: the left and bottom sides of the rectangle and the collinear edge of the PCB. The circular tracking algorithm tracks the line in a clockwise direction and thus follows the (incorrect) PCB edge line instead of turing the corner and following the left

side of the rectangle. The left side of the rectangle is isolated by this, and the line is deleted as its size falls below the threshold for a significant feature. In this example a size threshold of 30 pixels was used. The side can be retrieved by lowering the threshold, which would however include other small and insignificant line features, which complicates the feature matching stage. In the last sub-image of feature a) the result of pixel editing followed by re-vectorizing the feature can be seen. The pixels linking the PCB edge to the rectangular feature have been removed and the left side of the rectangular feature has been completed. The exact location of the added edge pixels are not crucial as a high accuracy sub-pixel edge detection is performed after the feature extraction and feature editing steps have been completed.

In figure 4.50 b) the line following algorithm has merged the faint line right above the rectangular feature with the feature boundary. A similar problem arises in aerial images where there are line features such as roads close to a house for example. The last sub-image of feature b) shows the correctly vectorized feature boundary. The pixels linking the boundary of the rectangle with the line above the rectangle were deleted in the binary edge map followed by re-vectorizing the feature.

The example shown in figure 4.50 c) is similar in nature to the first example where a line feature in close proximity to the polygon feature has been merged with the polygon outline due to its close proximity. The incorrect merging of the line features is rectified by deleting the pixels linking the line feature to the rectangle.

Chapter 5

Feature Classification

In this matching scheme the main emphasis was put on the extraction, classification and matching of features. The feature matching scheme was approached from a Machine Vision point of view, which differs from traditional image matching schemes. In traditional "blind" image matching schemes in Digital Photogrammetry only raster-based information about the image intensity (greyscales) is used without any knowledge of the objects or structures that are present in the image. Obtaining vector based information about objects in the image through feature extraction followed by feature classification allows us to adopt an intelligent matching scheme which can isolate those features that are of most interest in the image. In an aerial image for example rooftops can be isolated and classified, or in an industrial scene specific industrial parts can be recognized.

In the literature many schemes for describing and interpreting the shape of a polygon feature or an arc are presented. Davis [22] argues that a simple closed curve must be described not only in terms of its breakpoints or angles but also in terms of its sides. He presents a hierarchial graph description of the sides of the contour and derives measures to interpret the quality of the angles and sides. Of the more well known feature descriptors used in image analysis are $\phi(s)$-*plots* (Schenk) [88] (Hussain) [58], *Chain Codes* (Rosenfeld and Kak) [85] (Kamgar-Parsi *et al*) [62], *Fourier Descriptors* (Haralick and Shapiro) [47] (Tseng and Schenk) [102], the *Hough Transform* (Ballard) [6] (Haralick and Shapiro) [47] (Hussain) [58]. Other descriptors used include the Centroidal Profile or $r(\theta)$-plot (Hussain) [58], *Moments* (Teh and Chin) [99] (Haralick and Shapiro) [47] (Hong and Tan) [53], *Walsh Descriptors* (Sarvarayudu and Sethi) [87] and properties of polygons such as *area, compactness, bounding rectangles, elongatedness* and the *symmetric axis* (Haralick and Shapiro) [47] (Kamgar-Parsi *et al*) [62]. A more recent descriptor or signature reported in the literature proposes that shape descriptors should not only be invariant under changes in translation, scale and rotation but should also have high *entropy* (Jin) [61]. The entropy of signatures can be defined as the information carried by the coefficients of the signatures. A signature using the Smallest Enclosing Circle (SEC) of a feature is proposed. A feature outline is mapped into a unique n-dimensional shape-space using the SEC and matched according to this mapping.

The basic criteria for the feature descriptor used in this thesis, in order of perceived importance were the following:

1. Invariance under rotation, scale and translation
2. Uniqueness
3. Ease of implementation and interpretation
4. Accuracy

Of these criteria *invariance* is the most important. Comparing an image to its corresponding image in a stereo pair, geometric changes in rotation, scale and translation occur. Feature vectors are used for feature matching and must therefore be invariant under these geometric transformations to ensure that the correct feature matches are found.

The criteria of *uniqueness* specifies that a feature descriptor should uniquely define a specific feature. This criteria examines the applicability of a feature descriptor to image matching. A particular feature or type of feature should be uniquely described to distinguish different features during image analysis and subsequent image matching.

The *ease of implementation* is a valid criteria, especially from a computational point of view. In a complex image a lot of features have to be classified and the classification data stored. Having a feature descriptor that is relatively simple to implement that immediately extracts the vital information about a particular feature is essential in the fast and simple interpretation of the feature vector. This is equivalent to the criteria set by Jin [61] that feature descriptors should have high entropy.

Before the final high accuracy point matching is applied the accuracy is not of primary importance. The feature descriptor should be accurate enough to uniquely describe that particular feature, but some margin of error is allowed for at this stage. Feature matching is performed in two stages, of which feature descriptors are directly used only in the first stage, leaving some margin for error.

In this chapter different feature descriptors will be discussed. The $\phi(s)$ plot will be discussed first, with specific reference to the detection of circles and corners using the first derivative of the $\phi(s)$ plot, the $d\phi(s)$ plot. A new feature descriptor called the "Gradient-s" or G(s) plot will be discussed next, with an analysis of how its first derivative function dG(s) can be combined with the $d\phi(s)$ plot to simplify the corner detection process. Next the Förstner Interest Operator will be discussed, also with reference to how this operator can be combined with the $d\phi(s)$ plot to simplify the corner detection process. Finally other commonly used feature descriptors are briefly discussed, namely Chain Codes and Symbolic Representations of features, Centroidal Profiles, Fourier Descriptors, the Hough Transform and Higher Order Moments.

Feature vectors used in this matching scheme include the following:

- $\phi(s)$ plots and statistics
- $gradient(s)$ plots and statistics
- Boundary size

- Number of and positions of corners
- Type - Arc,Polygon,Circle

Note that parameters such as the number of corners and the type of feature depend on the feature descriptors such as the $\phi(s)$ plots and $gradient - s$ plots. Analysis of these plots or linear combinations of these plots allows us to find the number of and positions of corners in a feature.

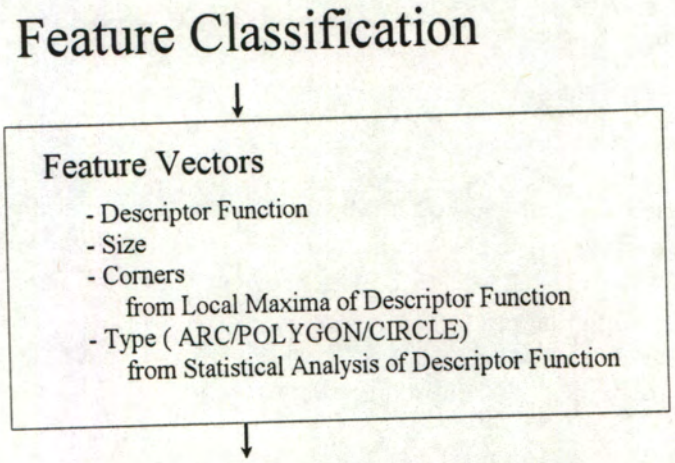


Figure 5.1: Feature Classification Flowchart

In this thesis project the first derivative of the $\phi(s)$ plot, the $d\phi(s)$ plot, is used as feature descriptor. Circles and near-circular features are detected using a statistical analysis of the $d\phi(s)$ plot. Corner points are detected by searching for local maxima in a feature descriptor that is formed by the product of the $d\phi(s)$ plot and the dG(s) plot, the first derivative of the "Gradient-s" function. The number of corners and type of feature obtained from the feature analysis are subsequently used during the matching stage.

5.1 PHI - S Plots

The $\phi(s)$ plot of a feature outline can be used to find the "corners" or breakpoints in the feature outline. Using the $\phi(s)$ plot as a "signature" to match the features simplifies the matching problem from a 2-D matching problem to a 1-D matching problem [88].

The $\phi(s)$ function plots the tangential direction of a curve as a function of the distance travelled around the curve. It indicates abrupt changes of direction as breakpoints in the curve. It is these breakpoints that indicate a corner or discontinuity in the feature outline (figure 5.2).

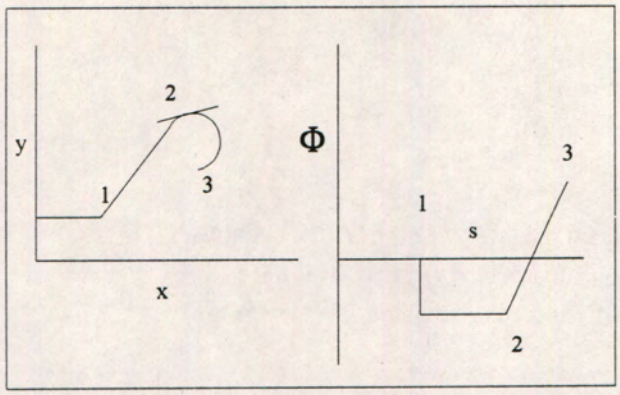


Figure 5.2: $\phi(s)$ plots of simple 2-D arcs

An arbitrary starting position is chosen, and the tangential direction is found at each point along the feature outline. Due to the discrete nature of the feature boundary, a small arc length, typically 3 pixels, is chosen to find the correct tangent. The direction is then found using the backward difference i.e. if the boundary vector can be defined as $S[n]$: where $n = 0..N-1$ then $\phi[n] = dirn(S[n], S[n-3])$ where "dirn" means the direction between S[n] and S[n-3] for an arc length of 3. (see figure 5.3).

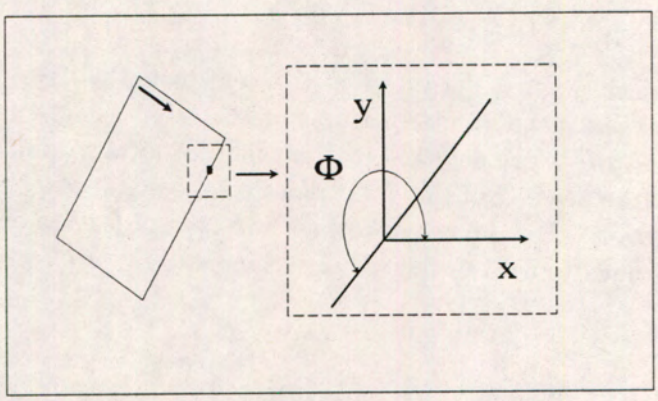


Figure 5.3: Generation of "$\phi(s)$" plots

The breakpoints in the $\phi(s)$ plots are easier to detect by taking the first derivative of the plot, the $d\phi(s)$ plot. As can be seen from figure 5.4 the corner points are now represented by prominent peaks. From a computational point of view it is now simpler to perform some smoothing on the signal, apply a threshold and then find the local maxima points than it is to find breakpoints directly from the $\phi(s)$ plot.

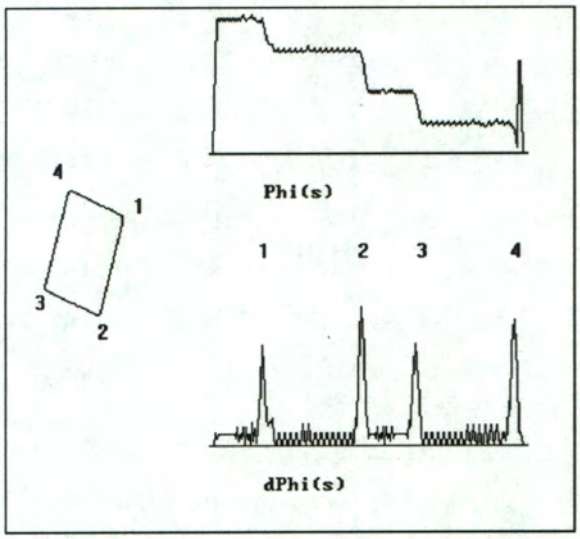


Figure 5.4: $\phi(s)$ and $d\phi(s)$ plots for a rectangle

Analysis of the $\phi(s)$ and $d\phi(s)$ plot shows that the height of the local maxima point of the $d\phi(s)$ plot is directly proportional to the angle of the corner represented by that local maxima. Refer to figure 5.5 for an analysis. In figure 5.5 a) the discrete feature boundary points at a corner are shown with the small line-segments of length L used to calculate the tangential directions.

Refer to figure 5.5 b) for the $\phi(s)$ plot of the feature around the corner and c) for the $d\phi(s)$ plot of the same region, calculated as $d\phi(S) = \phi(S + L) - \phi(S)$. The following parameters can be defined:

H_θ	=	The height difference $H_1 - H_2$ in the $\phi(s)$ value when going around a corner, equal to the angle of a corner
$H_{d\theta}$	=	The height of the peak of the $d\phi(s)$ plot representing the corner.
L	=	The length of the line-segments used to calculate the $d\phi(s)$ plot.
w	=	The width of the peak in the $d\phi(s)$ plot representing the corner.

The *transition band* is the distance in the s -domain during which the $\phi(s)$ plot changes from value H_1 to H_2, and it is equal to the peak-width w. The height $H_{d\phi}$ is calculated from the slope of the $\phi(s)$ curve in the transition band, calculated as H_ϕ/w. As w is a constant the height of the peak $H_{d\phi}$ is proportional to H_ϕ. If there is a large difference between the directions H_1 and H_2 the corner is "sharp" and $H_{d\phi}$ will be large. A "weak" corner on the other hand will be represented by only a small change in direction from H_1 to H_2 due to the flat angle and result in a small peak in the $d\phi(s)$ plot.

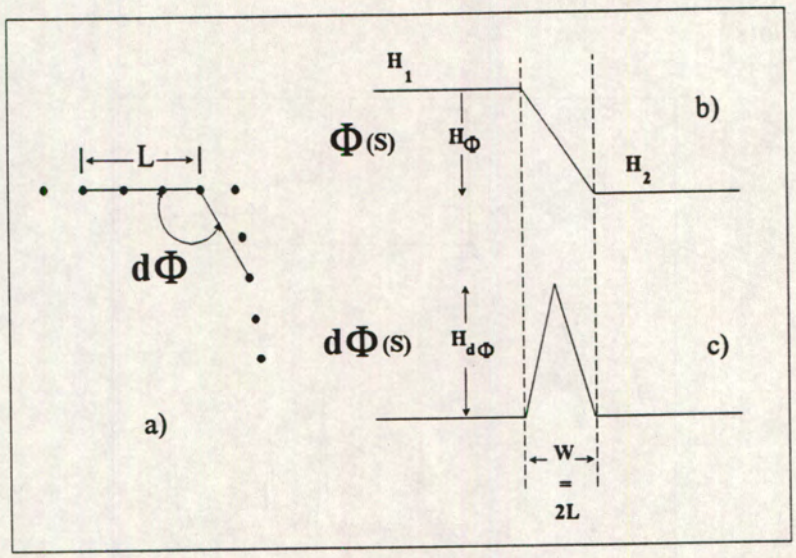


Figure 5.5: Analysis of the $\phi(s)$ and $d\phi(s)$ plot for a corner point

The $d\phi(s)$ plot is invariant under rotation and translation, and the basic form of the function is invariant to scaling. The scale of the feature just stretches or compresses the function along the "s" axis corresponding to the distance travelled along the feature outline.

5.1.1 Corner Detection Using $d\phi(s)$ Plots

Detecting all the maxima points of the $d\phi(s)$ plot indicates the number and location of all the corners on the feature boundary. The number of corners is used as one of the feature descriptors. The $d\phi(s)$ plot is first smoothed by a low-pass filtering function and then thresholded to find only prominent local maxima points.

The smoothing function applied in this thesis was a combination of three local averaging filters added in parallel.The local averaging filter replaces the function value with the average of the surrounding values, and can be expressed as

$$d\phi_{lpf}[n] = \frac{1}{m+1} \sum_{-m/2}^{m/2} d\phi[n+m] \tag{5.1}$$

where $d\phi[n]$ is the function value at n and m is the integer size of the local averaging window. In this thesis the results of local averaging filters with $m = 2$, 4 and 6 were added. Any other form of smoothing such as a gaussian smoothing function can also be applied.

The threshold used to detect only prominent local maxima points can be calculated in a number of different ways. One approach is to find the absolute maxima $d\phi_{max}$ of the smoothed $d\phi(s)$ plot $d\phi_{lpf}$ and set the threshold $d\phi_t$ at a certain fraction of this value e.g.

$$d\phi_t = 0.5 d\phi_{max}$$

The chosen threshold is crucial, as the number of corners of a feature is a vital input to the feature matching stage. If the threshold is set too low, weak corners that might not appear in a matching image or noisy "corners" might be detected. If the threshold is set too high, not all the prominent corners will be detected. Both these cases will hamper the chances of a correct match between the features being found.

To use all the information available in setting the threshold, a more robust statistical analysis of the smoothed $d\phi(s)$ plot is done. The new threshold is set using the average μ and standard deviation σ where

$$\mu = \frac{1}{N} \sum_{n=0}^{N-1} d\phi_{lpf}[n] \tag{5.2}$$

and

$$\sigma = \frac{1}{N-1} \sum_{n=0}^{N-1} (d\phi_{lpf}[n] - \mu)^2 \tag{5.3}$$

where N is the length of the feature border.

The threshold $d\phi_t$ is set using a linear combination of μ and σ

$$d\phi_t = \mu + k\sigma \tag{5.4}$$

with k a real number.

Experiments have shown that a convenient value of $k = 1$ gave better results than setting the threshold value at a fraction of the absolute maxima value.

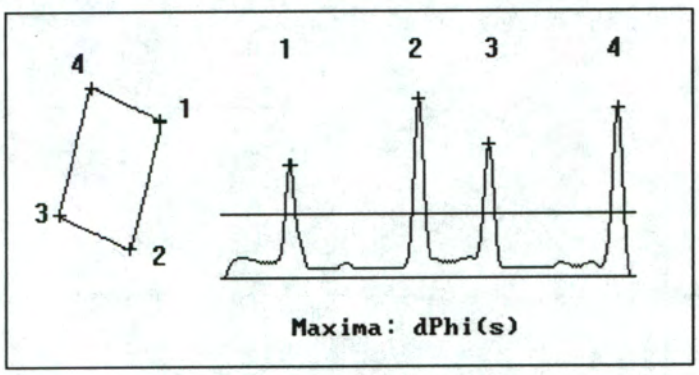


Figure 5.6: Corner detection from $d\phi(s)$ plots for a rectangle

Refer to figure 5.6 for an example of the corner detection using the smoothed $d\phi(s)$ plot. The threshold value $d\phi_t$ is shown, and the local maxima points and corresponding corner positions are indicated by crosses. Close scrutiny of corner [1] shows that two local maxima points are

indicated right next to each other. This is due to a subtle artifact caused by the local maxima finding algorithm, which reads something as follows:

```
if(dφ_lpf[n] > dφ_t)
                {
                if((dφ_lpf[n] ≥ dφ_lpf[n − 1]) and (dφ_lpf[n] ≥ dφ_lpf[n + 1]))
                                        Mark as a Local Maxima
                }
```

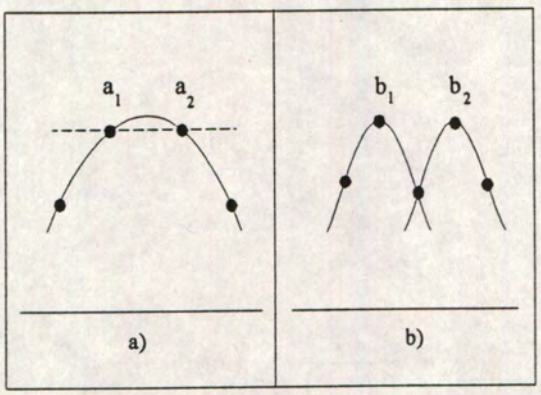


Figure 5.7: Equal local maxima a) and Local maxima in close proximity b)

In the specific case of corner [1] both points marked are at exactly the same level, and the "*less than/greater than or equal to*" operators select both points, marked as $a1$ and $a2$ in figure 5.7 a). If only the "*greater than*" and "*less than*" operators were used then neither point would be selected. With both points selected a post-processing function can discard one at a later stage. A similar problem arises when two corners are located very close to each other. If the smoothing function is not wide enough to smooth out the dip between the peaks then two local maxima will be found very close to each other, marked as $b1$ and $b2$ in figure 5.7 b). If the smoothing function has a wide enough window it will merge the two peaks and mark the corner at the local maxima of the new merged peak, which should be located somewhere in between the original corners. The corner marked as (3) in figure 5.8 shows an example of two peaks being merged by the smoothing function.

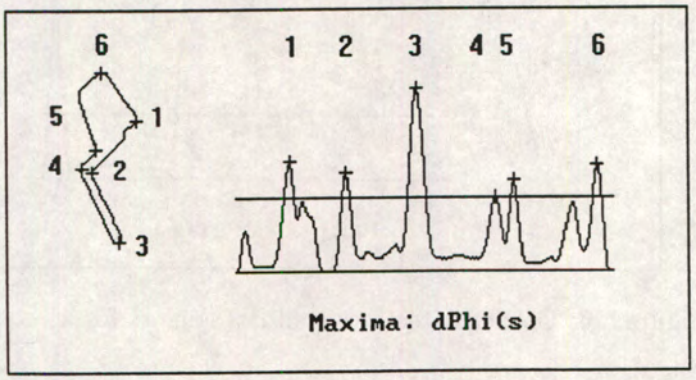


Figure 5.8: Corner detection from $d\phi(s)$ plots for a polygon

5.1.2 Circle Detection Using $d\phi(s)$ Plots

The presence of well defined geometric features in an image can be used to find cultural features such as buildings. Line features that can often be found in close-range and aerial images are circular and elliptic features. The most commonly used operator to detect circles and ellipses is the *Hough Transform*, which will be discussed in detail in section 5.4.4. An alternative method of detecting circular and elliptic functions is a Least-Squares geometric fit. The boundary coordinates of a feature can be fit to for example an ellipse function using a Least-Squares model. The resultant standard deviation of the adjustment will then indicate if the feature fits the ellipse function or not.

In this thesis project a simple measure based on $d\phi(s)$ plots has been devised to indicate the presence of a circular or near-circular feature. Statistical analysis of the $d\phi(s)$ plot can indicate the presence of a circular feature. This measure can then be used to classify a closed line-feature as a circle.

Refer to figure 5.9 for the $d\phi(s)$ calculation for an ideal circle. The direction $\phi(s)$ is calculated as the direction of the line of length L connecting point $[S]$ to point $[S-L]$. Similarly, the direction $\phi(S+L)$ is calculated as the direction of the line connecting point $[S+L]$ to point $[S]$. The first derivative function $d\phi(s)$ is now calculated as the difference function i.e. $d\phi(s) = \phi(S+L) - \phi(S)$.

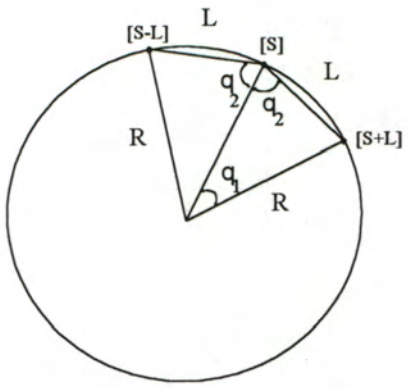


Figure 5.9: Calculation of $d\phi(s)$ for an ideal circle

The rotation angle θ_1 in radians, which is subtended by the line segment L, can be approximated for small angles as

$$\theta_1 \approx \frac{L}{R} \tag{5.5}$$

where L is the length of the line segment used to calculate the $d\phi(s)$ plot and R is the radius of the circle.

By applying the *Sine-Rule* for the triangle the angle θ_2 can be calculated from

$$sin\theta_2 = R\frac{sin\theta_1}{L} \tag{5.6}$$

The function $d\phi(s)$ is calculated from the change in direction between the line segments of length L. For a circle this function evaluates to

$$d\phi(s) = 2\theta_2 \tag{5.7}$$

From equations 5.5 and 5.6 it can be seen that the angle θ_2 depends purely on the constants L and R. The function $d\phi(s)$ for an ideal circle is thus constant around the whole circle boundary.

In this thesis project circular features were indicated by performing a statistical analysis of the $d\phi(s)$ plots to determine how much the value $d\phi(s)$ changes around the feature outline. If the changes are small it implies that the value $d\phi(s)$ is approximately constant, which indicates a circular feature.

To quantify the changes in the $d\phi(s)$ plot the average μ and standard deviation σ are calculated

$$\mu = \frac{1}{N}\sum_{n=0}^{N-1} d\phi[n] \tag{5.8}$$

and

$$\sigma^2 = \frac{1}{N-1}\sum_{n=0}^{N-1}(d\phi[n] - \mu)^2 \tag{5.9}$$

where N is the length of the feature border.

The deviations from the mean value μ are quantified by the standard deviation σ. To normalize these deviations the quotient of the two values is calculated.

$$\eta = \frac{\sigma}{\mu} \tag{5.10}$$

For a circular feature the standard deviation should be small as most of the values of $d\phi(s)$ should be close to the average value μ, resulting in the quotient η being small. Note that due to random noise present in the image and the discrete approximation of a circle in an image, the standard deviation will never be exactly zero. For a polygon or line feature with distinct corners the $d\phi(s)$ plot will show prominent peaks, resulting in a larger standard deviation and larger quotient η.

Refer to figure 5.10 for examples of the circle indicator quotient η for a circle, a triangle, a small square and an ellipse. The mean μ and standard deviation σ is shown on the $d\phi(s)$ plot for each feature.

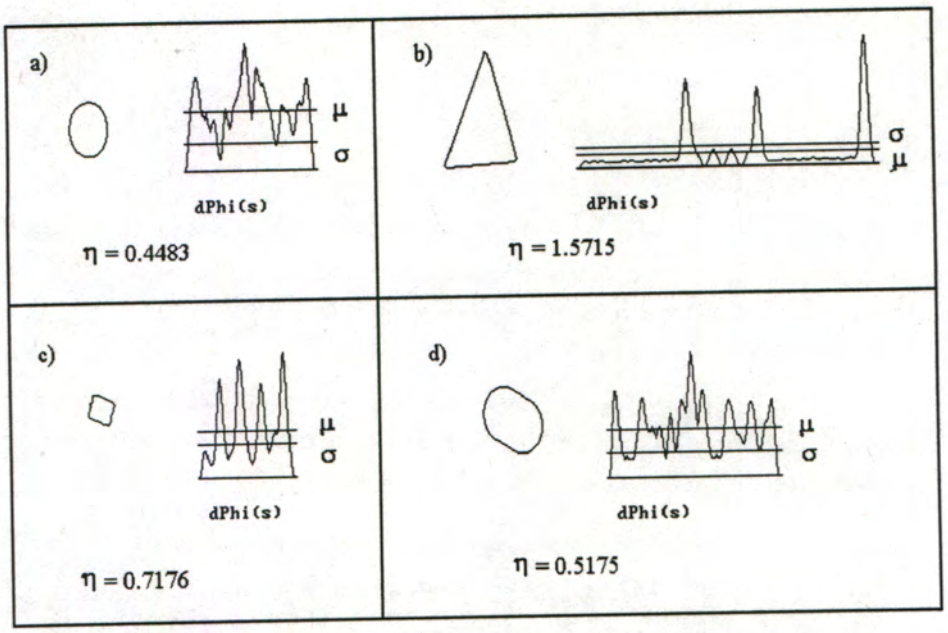


Figure 5.10: Circle indicator η for a circle a), triangle b), small square c) and ellipse d)

As expected the value η is relatively small for the circle and the ellipse shown in figure 5.10 a) and d) respectively. The quotient for the circle is 0.4483 and for the ellipse it is 0.5175. For the triangle shown in figure 5.10 b) the standard deviation σ is larger than the mean value μ of the $d\phi(s)$, which would indicate that η will be greater than one. In this case $\eta = 1.5715$, indicating a clear distinction between the triangle feature and the circular features.

The distinction between a general polygon feature with corners and piecewise smooth features such as circles and ellipses is not always that clear. Figure 5.10 c) shows an example of this occurring. In this thesis project a threshold value of η equal to 0.7 was used to distinguish between circular features and general polygons. The value η for the small square is 0.7176, which is very close to the chosen threshold value. Due to the discrete nature of the feature outline, a small square and a small circle will have a very similar outline, resulting in a very similar $d\phi(s)$ plot. The line segment of length L which is used for calculating the $d\phi(s)$ plot will be large relative to the feature boundary length, effectively smoothing the effect of the corner points on the $d\phi(s)$ plot.

In table 5.1 the quotients η for the features shown in the example in figure 5.10 above are compared to those obtained by performing sub-pixel edge detection on the feature boundary before analyzing the $d\phi(s)$ curve to find η.

Example	Feature	Pixel Boundary η	Sub-Pixel Boundary η
a)	Circle	0.4483	0.4718
b)	Triangle	1.5715	2.7617
c)	Square	0.7176	0.7248
d)	Ellipse	0.5175	0.6267

Table 5.1: Circle Indicators for Pixel and Sub-Pixel Feature Boundaries

From table 5.1 it should be noted that with more accurate edge coordinates the value η gives a clearer distinction between general polygon features and circular features. The values of η for the circular and near-circular features change only by small amounts, wheras the value of η for the triangle has increased by approximately seventy five percent, thus making it easier to distinguish between features.

Note that for a straight line the $d\phi(s)$ plot also yields a constant value, which is theoretically zero. The circle detector is only applied to closed-line features that have a mean $d\phi(s)$ value above zero to ensure that no open-line features or straight lines are classified as circles.

5.2 "Gradient-S" Plots

The tangential edge directions as described by the $\phi(s)$ plots are directly related to the directions of the edge gradients as calculated by the edge enhancement operator described in section 4.1 on edge enhancement. From figure 5.11 it can be seen that the tangential edge direction ϕ differs by ninety degrees from the edge normal direction θ_n.

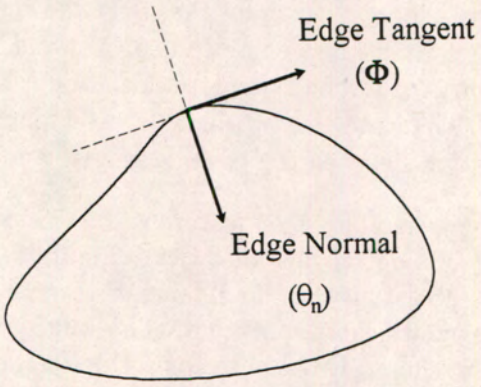


Figure 5.11: Tangential and Normal Edge directions on a feature boundary

The gradient normal θ_n to the tangential edge direction is calculated by using the edge gradients in the x- and y directions where

$$\theta_n = tan^{-1}(g_y/g_x) \tag{5.11}$$

and

$$g_1 = \sqrt{g_x^2 + g_y^2} \tag{5.12}$$

where g_x is the gradient in the x direction, g_y is the gradient in the y direction and g_1 is the resultant edge strength.

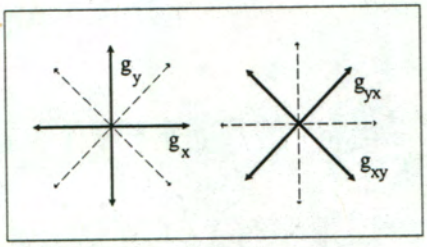


Figure 5.12: Basic gradient directions g_x, g_y, g_{xy} and g_{yx}

Note that this edge normal θ_n is an approximation to the actual edge normal as only two out of infinitely many edge gradients are taken to find the edge direction. For instance the same angle could be found by looking at the orthogonal set of gradients g_{xy} and g_{yx} that is rotated by $\pi/4$ with respect to g_x and g_y. The reader is referred to figure 5.12. The new edge normal direction would now be

$$\theta'_n = tan^{-1}(g_{yx}/g_{xy}) - \pi/4 \tag{5.13}$$

and the resultant edge gradient

$$g_2 = \sqrt{g_{xy}^2 + g_{yx}^2} \tag{5.14}$$

The two edge normals θ_n and θ'_n should be the same, but a change in the edge direction will be indicated by a difference in the magnitudes and the directions indicated by the resultant gradients g_1 and g_2.

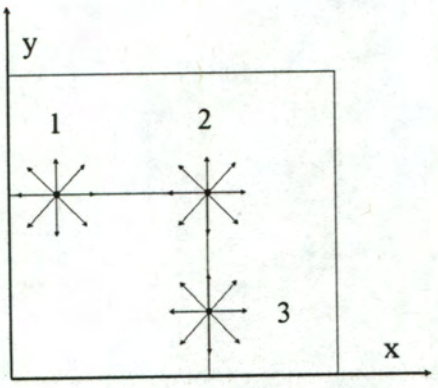


Figure 5.13: G(s) Calculation for a Corner Point

This can be seen through the simple example depicted in figure 5.13. The figure shows a perfect corner with background values = 0 and foreground values = 100. Compare the

resultant values g_1 and g_2 from the four gradient values g_x, g_y, g_{xy} and g_{yx} for the positions 1 and 3 on the feature sides and position 2 at the feature corner.

The results are depicted in table 5.2

Position	1	2	3
g_x	0	-100	-100
g_y	100	100	0
g_{xy}	100	0	-100
g_{yx}	100	100	100
g_1	100	141	100
g_2	141	100	141
$g_2 - g_1$	41	-41	41

Table 5.2: Table of G(s) Calculation

It can be seen from this simple example that there is an abrupt change in the value of $G(s)$ when comparing $G(s)$ for a point on the straight edge such as (1) and (3) to a corner point (2).

The gradient difference function as a function of the distance travelled around the feature boundary can be defined as

$$G(s) = g_2(s) - g_1(s) \tag{5.15}$$

where s is the distance travelled around the feature boundary.

This function indicates a change in the edge direction as a change in magnitude, similar to the response of the $\phi(s)$ plot to a change in edge direction.

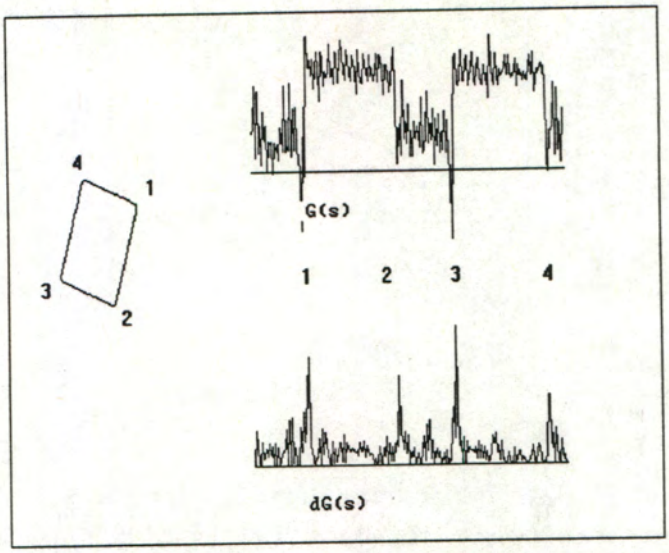


Figure 5.14: $G(s)$ and $dG(s)$ plots for a rectangle

The reader is referred to figure 5.14. The first derivative $dG(s)$ of the difference function $G(s)$ shows peaks at the corners similar to the $d\phi(s)$ plots. The local maxima of these peaks indicate the actual corner position.

The first derivative of the $G(s)$ function is calculated using the average difference gradient i.e.

$$dG(s) = \frac{G(s+1) - G(s-1)}{2} \tag{5.16}$$

In section 4.1.2 on edge detection the Canny edge operator was introduced. This gradient operator combines the functions of finding the gradient with smoothing by convolving the image with the first derivative of the Gaussian smoothing function. Figure 5.15 shows the resultant plot when the $dG(s)$ function is calculated from the convolution of the $G(s)$ function with the Canny operator. This operator can be used instead of first finding the gradient and then performing smoothing through local averaging.

In this instance

$$dG(s) = G(s) \otimes H(s) \tag{5.17}$$

where $\otimes$ represents convolution and $H(s)$ is the first derivative of the gaussian function.

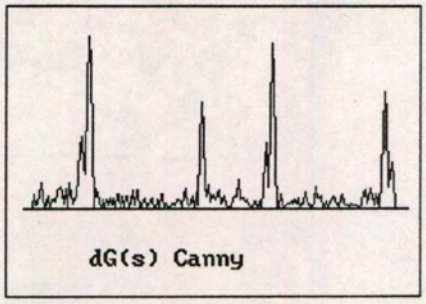


Figure 5.15: Calculation of $dG(s)$ from the Canny gradient operator

The function $G(s)$ is invariant under translation, but does depend on the rotation of the feature. The first derivative however only takes into the account the change in the greyscale difference $G(s)$ and is invariant under both translation and rotation. As in the case of the $d\phi(s)$ plot the basic shape of the function is invariant to scale, which only stretches or compresses the "s" axis.

As can be seen from figure 5.14, strong local maxima appear only at the corner positions and weaker local maxima appear at other points which do not necessarily indicate a significant corner point. To emphasize only significant corner points the $dG(s)$ plot is used to enhance prominent peaks in the $d\phi(s)$ plot. The product of the $dG(s)$ and the $d\phi(s)$ functions results in a plot which clearly enhances only prominent corner points and suppresses weak corner points. See figure 5.16.

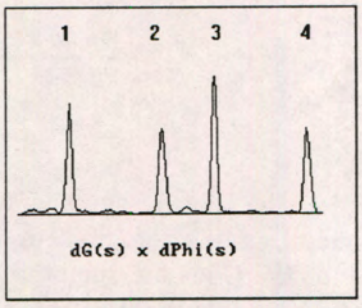


Figure 5.16: $dG(s)$ x $d\phi(s)$ plot for a rectangle

5.2.1 Corner Detection Using $dG(s)$ Plots

The resultant product function shown in figure 5.16 enhances prominent corners, thus making the corner finding operation less sensitive to the threshold value $d\phi_t$ as defined in equation 5.4.

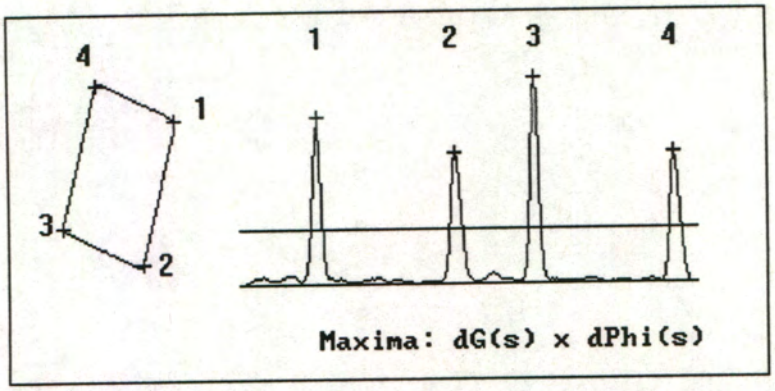


Figure 5.17: Corner detection from the $dG(s)$ x $d\phi(s)$ plot for a rectangle

In figure 5.17 the four corners have correctly been found by the corner detector. The corner marked (1) is now distinct and marked only once. Using only the $d\phi(s)$ plot this corner was marked as two corners right next to each other as shown in figure 5.7 a).

Note that the prominent peaks are higher relative to the threshold value than was the case for the corresponding peaks of the smoothed $d\phi(s)$ plot for the same rectangle.

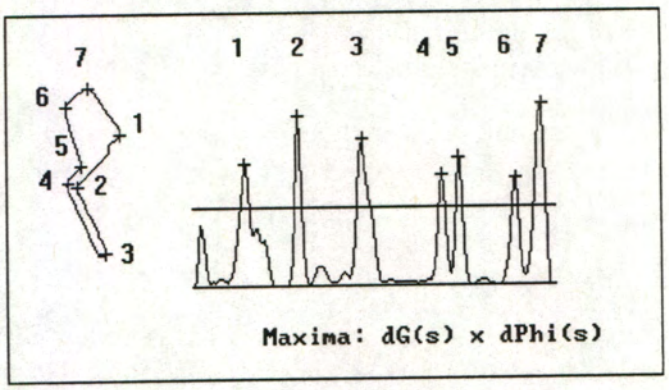


Figure 5.18: Corner detection from the $dG(s)$ x $d\phi(s)$ plot for a polygon

In figure 5.18 the function $dG(s)$ has enhanced peaks that were below the threshold value $d\phi_t$ in the $d\phi(s)$ plot to above the threshold. Another corner is thus found, marked as 6. This corner was not found using only the $d\phi(s)$ function.

5.3 Förstner Interest Operator

The Förstner Interest Operator is a well established operator that selects points of interest on the basis of the greyscale gradients in a window around the points. The Interest Operator has been designed for the following tasks:

- Least Squares Matching
- Corner Detection
- Weighted Centre of Gravity Detection
- Detection of the Centre of Circular Features

explained in more detail in Förstner [26].

Of particular interest is the corner detection application, which is equivalent to the weighted centre of gravity where the weights are defined by the greyscale gradients.

The interest operator is usually implemented over the whole image using a 2-step procedure:

1. Point detection by searching for optimal windows
2. Accurate point location within the selected optimal windows.

The normal equation system for a corner centered at point (x_0, y_0) in the image is

$$N \begin{bmatrix} y_0 \\ x_0 \end{bmatrix} = \begin{bmatrix} \sum g_y^2 y + \sum g_x g_y x \\ \sum g_x g_y y + \sum g_x^2 x \end{bmatrix} \tag{5.18}$$

where g_x and g_y are the greyscale gradients calculated using any of the gradient operators such as the gradient-sum box filter or the Canny edge operator. The sums are taken over the whole window around the point, typically an area of 5x5 to 7x7 pixels in size.

For the corner detection task the normal equation coefficient matrix takes the form

$$N = \begin{bmatrix} \sum g_y^2 & \sum g_x g_y \\ \sum g_x g_y & \sum g_x^2 \end{bmatrix} \tag{5.19}$$

The precision of an estimated corner point can be derived from the *size* and the *form* of the error ellipse.

The size of the error ellipse is reflected by the weighting function w where

$$w = \frac{1}{trN^{-1}} = \frac{detN}{trN} \tag{5.20}$$

where $detN$ is the determinant of N and trN is the trace of N.

The roundness of the error ellipse is reflected by the function q where

$$q = \frac{4detN}{tr^2N} \tag{5.21}$$

where the roundness measure q lies in the range between 0 and 1. If q = 1 the error ellipse is a circle and if q = 0 then the window lies on an ideal edge.

The values w and q can be used to select optimal windows in which to find interest points. The basic criteria to be met are

Criteria 1: The error ellipse should be small which implies w must be large

Criteria 2: The error ellipse should be close to a circle which implies q close to 1.

In traditional applications of the Förstner interest operator a weight is assigned to each window position by thresholding both w and q i.e.

$$w^*(x,y) = \begin{cases} w(x,y) & \text{if} \quad w(x,y) > w_t \\ & \text{and} \quad q(x,y) > q_t \\ 0 & \text{all other cases} \end{cases} \tag{5.22}$$

where the threshold values w_t and q_t can be critical and non-predictable.

In this investigation the thresholds were not calculated as the values of w and q were used only as corner indicators in conjunction with the $d\phi(s)$ plots.

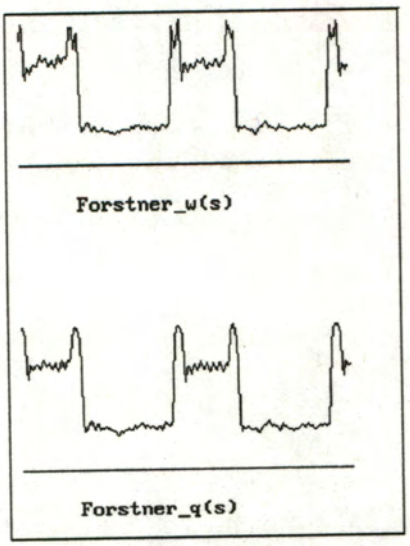


Figure 5.19: $w(s)$ and $q(s)$ plot for a rectangle

Observing the plots of $w(s)$ and $q(s)$ around a feature boundary where s is the distance travelled around the boundary, it displays similar characteristics to the $d\phi(s)$ and $dG(s)$

plots. At the corner points indicated by the numbers $1 \cdots 4$ the functions both peak, which can be used to find the corner position. See figure 5.19.

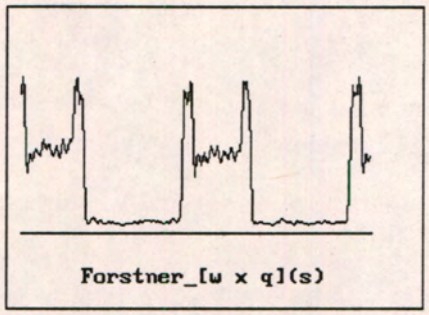


Figure 5.20: $w(s)$ x $q(s)$ plot for a rectangle

The plots of w and q must both peak at the same point, which would indicate the interest point. To take both plots into account simultaneously the product between the two is formed i.e. w x q, which is shown in figure 5.20.

This function can also be used as an "amplifier" function to enhance the corner peaks of the $d\phi(s)$ plot, much the same as the $dG(s)$ function was used for this purpose. Figure 5.21 shows the resultant plot for the same rectangle as used in figures 5.14 to 5.20.

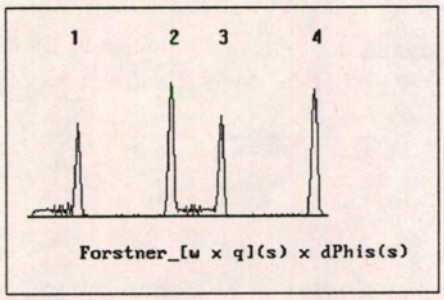


Figure 5.21: $[w(s)q(s)]$ x $d\phi(s)$ plot for a rectangle

The Förstner Operator has been designed to be invariant under geometric distortions such as rotation, scale and translation.

5.3.1 Corner Detection Using the Förstner Interest Operator

The resultant Förstner product function shown in figure 5.21 enhances prominent corners in a similar fashion to the $dG(s)$ function. This product function again makes the corner finding algorithm less sensitive to the chosen threshold value $d\phi_t$.

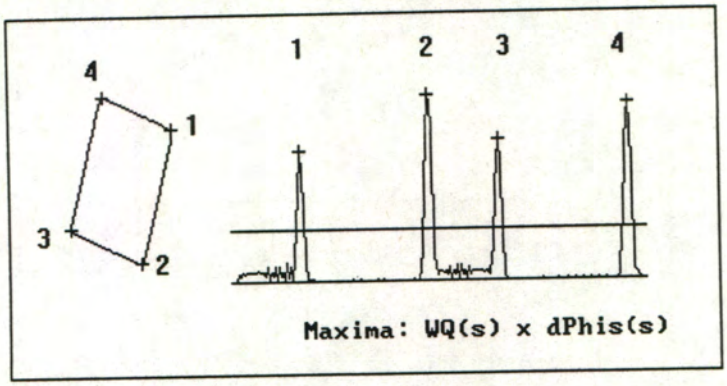


Figure 5.22: Corner detection from the $[w(s)q(s)]$ x $d\phi(s)$ plot for a rectangle

In figure 5.22 the four corners have correctly been found by the corner finding algorithm. The corner marked (1) is also distinct and marked only once. Note that the prominent peaks are again higher relative to the threshold value than was the case for the corresponding peaks of the smoothed $d\phi(s)$ plot for the same rectangle.

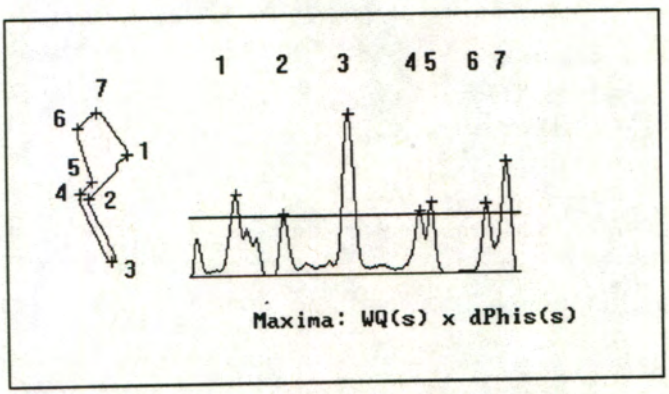


Figure 5.23: Corner detection from the $[w(s)q(s)]$ x $d\phi(s)$ plot for a polygon

In figure 5.23 the enhancement function $w(s)q(s)$ has enhanced peaks that were below the threshold value $d\phi_t$ in the $d\phi(s)$ plot to above the threshold. The same extra corner (6) as found using the $dG(s)$ plot as enhancement function is found. Note that in this plot the prominent peaks are dangerously close to the threshold value, which could mean that corners are not found in a corresponding feature. Ideally the corresponding corners should be found in all the images to ensure a successful feature match.

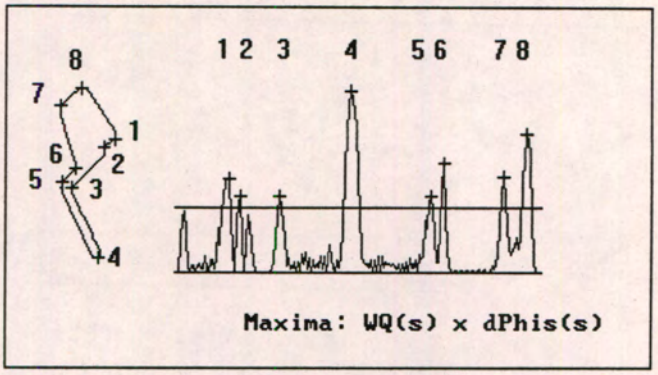


Figure 5.24: Corner detection from the unsmoothed $[w(s)q(s)]$ x $d\phi(s)$ plot for a polygon

In comparison, observe the effect of not applying the local averaging smoothing function before the corner detection. In figure 5.24 the corner marked (2) is detected, as the smoothing function has not merged the peaks in the close vicinity of the peak corresponding to corner 2. The smoothing is however deemed necessary as the prominent local maxima are kept while the weak and often spurious local maxima points are averaged out and discarded.

In this thesis project the corners were found using the $dG(s)$ function as enhancement operator for the corners. The $dG(s)$ plot is computationally a lot simpler to calculate than the Förstner Interest Operator, and was found to generally enhance prominent peaks more relative to the chosen threshold value.

5.4 Other Feature Descriptors

In the literature a number of other feature descriptors are discussed. Of the descriptors or one dimensional "signature" functions describing the shape of a line feature the Chain Code, Fourier Descriptors, the Hough Transform and Moments will be discussed.

5.4.1 Chain Codes and Symbolic Representation of Features

The Chain Code is one of the oldest feature descriptors in use and was used as early as in 1965 by Freeman for the segment fitting of curves using chain code correlation [62].

The chain code is a discrete version of the $\phi(s)$ plot that is quantized in increments of $\frac{\pi}{4}$ degrees. It is a directional code which describes an arc by a starting point and a sequence of directional moves. In moving from one point to the next around the feature border we can go to one of the eight neighbours of the starting position. The eight neighbours can be numbered as follows:

$$\begin{matrix} 3 & 2 & 1 \\ 4 & * & 0 \\ 5 & 6 & 7 \end{matrix}$$

Each position around the central pixel marked $*$ is thus defined by one of the octal digits 0..7 with each increment of the chain code representing a change of $\frac{\pi}{4}$ in direction [85].

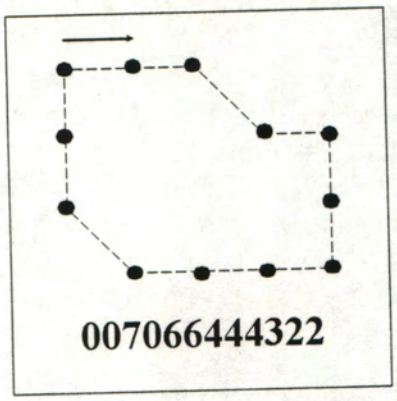


Figure 5.25: A simple chain-coded feature

Refer to figure 5.25 for an example of a simple closed feature with chain code 007066444322. The starting position is at the top, left corner of the feature and the feature is coded in the clockwise direction.

Two chain codes can be compared and a correlation value calculated to establish if the arcs represented by the chain codes match or not [62]. Given two chain code strings $A = a_1\ a_2\ ...\ a_n$ of length n and $B = b_1\ b_2\ ...\ b_m$ of length m where $m \leq n$ then the correlation function Φ_{AB} at position j is

$$\Phi_{AB}[j] = \frac{1}{m} \sum_{i=1}^{m} cos([b_i - a_{i+j}] mod_8 \frac{\pi}{4}) \tag{5.23}$$

If there is a matching position j i.e. where there is a 100 % correlation between A and B then $\Phi_{AB}[j] = 1$.

An extension of the chain code is to represent a feature outline as a string of line segments. Each descriptor of the sequence thus has two elements (a_i, l_i) where a_i is the angle of the i_th line segment and l_i is its length. The best match is found using a relaxation method [62]. Bhanu and Faugeras use a hierarchial approach to solve for the so-called *segment matching* problem [12]. Features are modelled as polygons and the line segment descriptors are matched by maximizing a criterion function based on the ambiguity and inconsistency of classification.

In general arcs can be represented as attribute strings and symbolic string matching can be used [62]. An example of such a symbolic string representation can be found in Liu [70]. The

curvature signature function of a closed feature is found by convolving the edge direction of each boundary point with the first derivative of the Gaussian function. This is effectively the same as the $d\phi(s)$ plot. This curvature function is now represented symbolically by quantizing the function into different levels. Within each quantization level a positive slope element is represented by the symbol "x" and a negative slope element by the symbol "y". This string representation describes the relative amplitude of the peaks and valleys on the signature function, and matching is performed by calculating the cost of transforming one string to another.

5.4.2 Centroidal Profile

A feature descriptor that is very similar to the $\phi(s)$ plot is the centroidal profile, commonly known as the $r(s)$- or $r(\theta)$- plot . The centroidal profile of a feature is obtained by plotting the vector length between the feature centre-of-mass and the feature boundary [58]. The vector length can be plotted as a function of either the angle θ or s, the length travelled along the feature boundary. Note that for any particular angle there can be more than one vector to the feature outline so plotting the $r(s)$ curve is preferred.

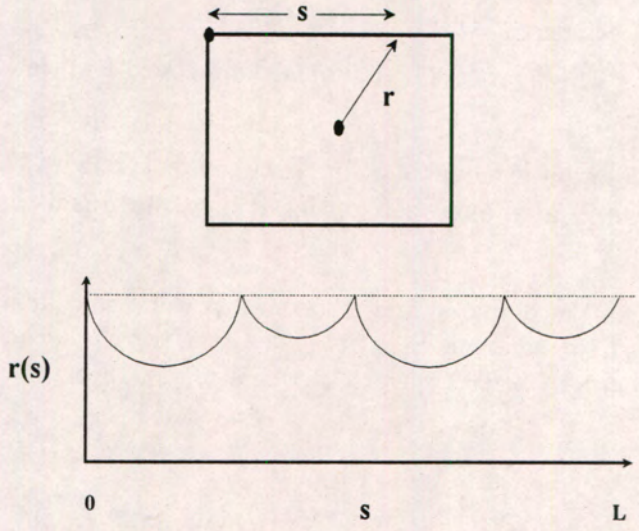


Figure 5.26: "r(s)" plot for a simple rectangular feature

The $r(s)$ plot for a simple rectangular feature is shown in figure 5.26. The distance r is measured from the feature centre-of-mass to the feature outline and the distance s is measured from the starting point at the top, left corner of the rectangle. Note that as with the $d\phi(s)$ and dG(s) plots the corner positions are indicated by the local maxima of the r(s) plot.

5.4.3 Fourier Descriptors

All of the feature descriptors described so far describe the feature outline as a function of one variable s. Based on Fourier Analysis, described in section 3.2, a "time-domain" signal can

be represented in the "frequency" domain by the Fourier Series expansion. The principle of Fourier Descriptors is to find the Fourier Series of a "time-domain" feature descriptor such as the $\phi(s)$ or $r(s)$ functions and use a finite number of the Fourier Series coefficients as Fourier Descriptors [47]. For continuity with the earlier discussion of the Fourier Series the "spatial-time domain" variable "s", the distance travelled along the feature outline, will be represented as the discrete-time variable "n".

From the Discrete Fourier Transform (DFT) described in section 3.2, the synthesis and analysis equations for the discrete function $x[n]$ of length n can be formed.

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j(2\pi/N)kn} \tag{5.24}$$

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn} \tag{5.25}$$

In this case $x[n]$ represents either the $\phi(s)$ or $r(s)$ functions. The DFT coefficients $X[k]$ represents the feature descriptor in the frequency domain and can be used as the Fourier Descriptors.

Tseng and Schenk [102] use Fourier Descriptors to describe the x and y positions of a feature boundary. The Fourier Descriptors are used for matching using a Least Squares Adjustment. Curves are treated as continuous, periodic functions. An open line or arc is made periodic by tracing and then retracing the line backward so that a closed boundary is obtained. As an example the case of a closed polygon is examined. Refer to figure 5.27 for an example of a closed polygon and its periodic representation.

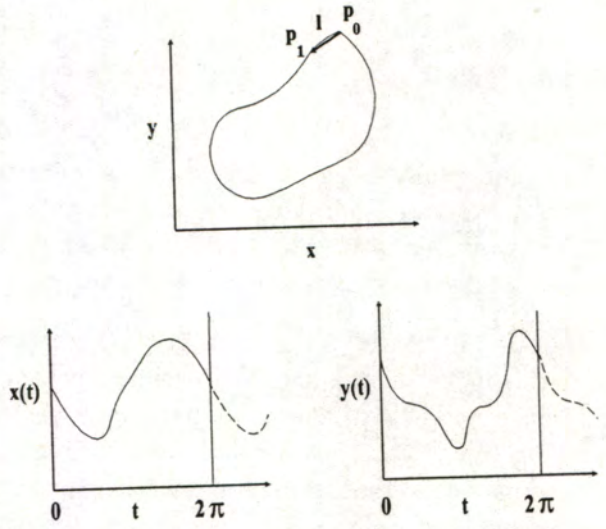


Figure 5.27: A 2-D closed polygon and its periodic position functions

A feature outline can be described by two periodic functions x(t) and y(t). The sample "t"

is defined as $2\pi l/L$, where L is the perimeter of the feature boundary and l denotes the arc length between samples. The Fourier Series expansion can be represented in matrix form as:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} a_0 \\ c_0 \end{pmatrix} + \sum_{n=1}^{\infty} \begin{pmatrix} a_n & b_n \\ c_n & d_n \end{pmatrix} \begin{pmatrix} cos(nt) \\ sin(nt) \end{pmatrix} \tag{5.26}$$

with:

$$\begin{aligned} a_0 &= 1/2\pi \int_0^{2\pi} x(t)dt & c_0 &= 1/2\pi \int_0^{2\pi} y(t)dt \\ a_n &= 1/\pi \int_0^{2\pi} x(t)cos(nt)dt & b_n &= 1/\pi \int_0^{2\pi} x(t)sin(nt)dt \\ c_n &= 1/\pi \int_0^{2\pi} y(t)cos(nt)dt & d_n &= 1/\pi \int_0^{2\pi} y(t)sin(nt)dt \end{aligned}$$

The translation parameters a_0 and c_0 are the mean values of $x(t)$ and $y(t)$ respectively and represent the centre-of-mass of the feature.

Taking the coefficients a_n, b_n, c_n and d_n of this function allows us to do a least squares matching using these frequency components. There are a finite number of these coefficients and these are used as the Fourier Descriptors.

5.4.4 The Hough Transform

The Hough Transform has successfully been used in many applications to find analytical curves such as straight lines, circles and parabolas in images (Hussain [58]),(Adamos and Faig [2]), (Argialas and Krishnamurthy [4]). The work of Ballard [6] generalized the Hough Transform to detect not only analytical curves but also more general shapes.

The Hough Transform is given the family of curves that is sought and produces the set of curves from that family that appears in the image. Consider an analytic curve of the form $f(u, a) = 0$ where u is an image point (x, y) and a is a parameter vector dependent on which family of curves we are trying to detect. The Hough Transform represents a transformation from two-dimensional image-space to n-dimensional parameter space, where n is the number of free parameters representing an analytic curve. Each neighbourhood of the image transformed will map to a point or a set of points in the Hough parameter space. See table 5.3 for the parameters and governing equations of some common analytical curves detected using the Hough transform.

Analytic Form	Parameters	Equation
Line	R, θ	$x\,cos\theta + y\,sin\theta = R$
Circle	x_0, y_0, R	$(x - x_0)^2 + (y - y_0)^2 = R^2$
Parabola	x_0, y_0, R, θ^*	$(y - y_0)^2 = 4R(x - x_0)^2$
Ellipse	x_0, y_0, a, b, θ^*	$\frac{(y-y_0)^2}{a^2} + \frac{(x-x_0)^2}{b^2} = 1$

Table 5.3: Analytic Curves With Parameter Vectors and Governing Equations

Note that for both the parabola and the ellipse the rotation angle θ must be taken into account as well.

The Hough Transform tries to find the locus in parameter space that will define the curve at a specific point. If for example we are trying to find circles in an image, we would look at possible edge pixels i.e. pixels with a high gradient magnitude and ask the question:" *If this pixel is to lie on a circle, what is the locus for parameters of that circle?*". Figure 5.28 shows the locus of parameters for a circle in 3-D Hough space with variables x_0, y_0 and R. For a circle the parameter locus is represented by a right circular cone.

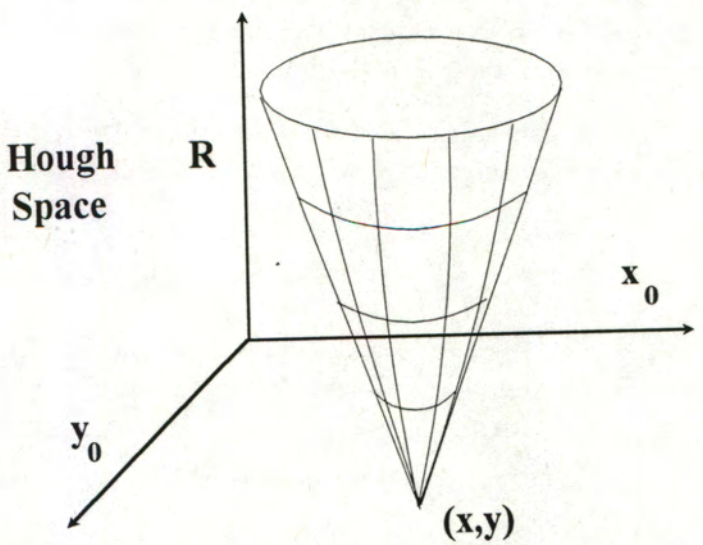


Figure 5.28: Locus of parameters for a circle in Hough-space

In order to reduce the search space the first derivative of the function $f(u, a)$ is taken into account. This expression must also evaluate to zero i.e.

$$f(u,a) = 0 \quad \text{and} \quad \frac{\partial f}{\partial u}(u,a) = 0$$

This introduces a term dy/dx which is known from the edge directions on the feature boundary since

$$\frac{dy}{dx} = tan[\phi(u) - \frac{\pi}{2}] \tag{5.27}$$

where $\phi(u)$ is the edge gradient direction.

In the example of searching for circles the addition of the gradient direction information will decrease the search locus from a right circular cone to a line in Hough space.

The Hough algorithm for analytic curves can thus be summarized as follows:

1. For a curve $f(u,a) = 0$ form an accumulator array $A(a)$ initially set to zero.
2. For each edge pixel u compute all a such that $f(u,a) = 0$ and $\frac{\partial f}{\partial u}(u,a) = 0$.
3. Increment the corresponding accumulator array entries $A(a) = A(a) + 1$
4. After all the edge pixels have been processed local maxima in the accumulator array A correspond to curves of f in the image.

The local maxima of the accumulator array tells us what combinations of the curve parameters are found in the image, but not exactly where they are. To determine the actual positions of the curves an additional array can be constructed which keeps track of which image points contribute to which entries in the accumulator array.

For the parabola and the ellipse in table 5.3 the rotation angle θ can be taken into account by rotating the x and y parameters by θ and adding the rotation to the dy/dx term i.e.

$$\frac{dy}{dx} = tan[\phi(u) - \theta - \frac{\pi}{2}] \tag{5.28}$$

Ballard [6] generalizes the Hough Transform to detect non-analytic curves. He defines a generalized shape with the following parameter vector:

$$a = \{x_0, y_0, s_x, s_y, \theta\}$$

where (x_0, y_0) is a reference origin for the shape, s_x and s_y are scale factors in x and y respectively and θ is a rotation angle. The variable distance r between the reference point and the boundary points are recorded and stored in an R-table as a function of the angle θ i.e $r(\theta)$. For each edge pixel u in the image the corresponding points $u + r(\theta)$ are incremented in the accumulator array. Maxima in the accumulator correspond to possible instances of the general shape in the image.

5.4.5 Shape Analysis using Moments

Moments and functions of moments have been used to recognize image patterns and general two-dimensional objects [47] [99] [53] [64]. For moments to be used successfully in feature recognition nonlinear combinations of moments are formed, called *Moment Invariants.* Moment invariants are moments that are invariant under geometric transformations such as rotation, scale and translation. Hong and Tan [53] use moments to transform point sets into canonical forms that can be used for matching purposes. Teh and Chin [99] give a review of moment based image analysis methods. The moments they consider include Geometric Moments, Legendre Moments, Zernike Moments, Pseudo-Zernike Moments, Rotational Moments and Complex Moments.

Haralick and Shapiro [47] trace the history of the use of geometric moments in image analysis.

For a general two-dimensional shape S, the $(j,k)_{th}$ moment is calculated from the boundary points (x,y) as:

$$M_{jk}(S) = \sum_{(x,y)\in S} x^j y^k \tag{5.29}$$

The *centre-of-mass* (x_0, y_0) of S can be calculated in terms of moments as

$$x_0 = \frac{M_{10}}{M_{00}} \tag{5.30}$$

$$y_0 = \frac{M_{01}}{M_{00}} \tag{5.31}$$

Using this centre-of-mass the *central* $(j,k)_{th}$ *moment* of S can be defined by

$$\mu_{jk} = \sum_{(x,y)\in S} (x - x_0)^j (y - y_0)^k \tag{5.32}$$

The central moments are invariant under *translation.*

The *standard deviation* can be expressed in terms of geometric moments as:

$$\sigma_x = \sqrt{\frac{\mu_{20}}{M_{00}}} \tag{5.33}$$

$$\sigma_y = \sqrt{\frac{\mu_{02}}{M_{00}}} \tag{5.34}$$

These standard deviations can be used to normalize the image coordinates s.t.

$$x' = \frac{(x - x_0)}{\sigma_x} \tag{5.35}$$

$$y' = \frac{(y - y_0)}{\sigma_y} \tag{5.36}$$

These normalized coordinates (x', y') thus have zero means and standard deviations of 1. The *normalized moments* can now be defined by

$$m_{jk} = \frac{\sum (x')^j (y')^k}{M_{00}} \tag{5.37}$$

These normalized moments are invariant under translation and different scales in x and y.

In order to obtain invariance under rotation the *normalized central moments* are defined as

$$\eta_{jk} = \frac{\mu_{jk}}{\mu_{00}^{\gamma}} \quad \text{where} \quad \gamma = \frac{j+k}{2} + 1 \tag{5.38}$$

From these normalized central moments seven functions that are rotation invariant can be defined:

$$\phi(1) = \eta_{20} + \eta_{02}$$

$$\phi(2) = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi(3) = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi(4) = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\begin{aligned}\phi(5) = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ & (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned}$$

$$\begin{aligned}\phi(6) = {} & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \\ & 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned}$$

$$\begin{aligned}\phi(7) \quad & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\ & (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned}$$

The last function, $\phi(7)$ is also invariant to shear.

Chapter 6

Feature Matching

Most of the feature descriptors discussed in chapter 4 have been used not only in monocular image analysis applications but also in image matching applications. In this thesis project the extracted features were used for image matching, therefore the criterion of invariance to geometric distortions was the most important in deciding on which feature descriptor to use.

The *Förstner Interest Operator* has been used successfully by Zong *et al* in an automatic image orientation system [107]. Point features are extracted using the Förstner Interest Operator and corresponding points indicated by the operator are matched using area correlation. In other feature matching schemes extracted line features have been used extensively as features for image matching. The $\phi(s)$ plots of line features have successfully been used for feature matching by researchers at the Ohio State University amongst others. Stereo image matching using line features described by $\phi(s)$ plots is discussed by Schenk *et al* [89], Stefanidis *et al* [95], Toth and Schenk [100] and Greenfeld and Schenk [33] amongst others. Matching open and closed line features using *Fourier Descriptors* is discussed by Tseng and Schenk [102]. Line features are described using Fourier Descriptors and these descriptors are matched using a Least-Squares approach. Adamos and Faig use the *Hough Transform* to detect targets in digital photogrammetry [2]. Zong *et al* apply the Hough Transform as a "figural continuity" constraint in an image matching scheme based on zero-crossing line segments [107]. The figural continuity criterion implies that the disparity values along zero-crossings must be continuous. Matched points belonging to a zero-crossing line segment are transformed into Hough space. Continuous disparity values show up as clusters in Hough space and the size of these clusters is used to determine correspondence. Geometric *Moments* are used as features in a point pattern matching technique described by Hong and Tan [53]. Point sets are transformed to canonical forms under affine transformation using moments and matched by directly comparing the resultant canonical forms.

In the literature there are numerous other examples of feature based matching techniques using point, line and region features. Crombie presents a conceptual analysis of feature extraction for image analysis and -matching [21]. He proposes that the existence of structure in an image can be indicated by features such as point-density data, texture, edges and existing cartographic knowledge. These features can be combined and organized to more completely describe points in an image. One of the first successful feature based matching techniques

using line features is described by Baker [5]. Edges are extracted at different resolution levels and matched using a "coarse to fine" implementation of a dynamic programming matching algorithm. Dynamic programming refers to a matching method that optimizes a cost function that takes the similarity of features into account. Baker proposes that the *significance* of features should be determined and used to first match significant features. Features are deemed significant if they form part of a meaningful structure such as a rooftop or a road. Grimson analyses a feature based stereo matching system first proposed by Marr and Poggio, based on aspects of the human vision system [36]. This system is similar to the feature matching scheme discussed by Baker [5]. Line features are found by identifying the zero-crossings of an image convolved with the Laplacian-of-Gaussian function. Rectified images are matched and the matching algorithm searches for corresponding zero-crossing points in each scanline using a "coarse to fine" iteration to search through zero-crossings obtained from multi-resolution representations of the input images. A similar line matching approach is used by Ohta and Kanade [80]. Matched line points along scanlines of rectified images are also used, but an *inter-scanline* search is introduced to check the dependency of corresponding edge points on the corresponding points found in the neighbouring scanlines. Costabile *et al* discuss a shape matching scheme that is applied to track shapes from one frame to another in time-varying images [18]. Closed line features are decomposed into convex parts using a polygonal approximation of the feature boundary. Each convex line segment is assigned attributes and segments are matched by applying a tree search that minimizes a difference function between likely matching candidates.

Benard presents a feature matching scheme based on feature extraction and dynamic programming [9]. Rectified images are used as input to the matching scheme. Each scanline is described by a set of objects such as pixels, edges, blobs, vectors etc. and descriptions are matched using a dynamic programming algorithm known as the Vitterbi algorithm. Corresponding objects are matched by applying special rules set out in a grammatical form. Edge segments are used by Medioni and Nevatia as features in an image matching scheme [76]. Extracted line segments are described according to a) the coordinates of the end points, b) orientation of the line and c) the strength of the line, defined by the average gradient contrast. Candidates for matching are obtained from these descriptions and matches are verified using a "minimal differential disparity" criterion. El-Hakim describes an image matching system which extracts two-dimensional "blob" features used for matching[24]. Images are segmented using tesselation, resulting in white "blobs" on a black background. Blobs are labelled and known target points are recognized from these blobs. Control points are used to perform an exterior orientation and the exact centres of the blobs are calculated to sub-pixel accuracy, increasing the accuracy of the exterior orientation. Greenfeld and Schenk [33] present an alternative line matching scheme to the method of using $\phi(s)$ plots as previously employed by Schenk *et al* [89]. Line features are approximated using a polygon approximation which links breakpoints (points of high curvature) with straight lines. The breakpoints become vertices of the polygon approximation, which are used as primitives for matching. Further results and notes on the practical application of this method are presented by Greenfeld *et al* [34].

Horaud and Skordas were one of the first groups to recognize the importance of not only the local structure of a line feature but also the relationship between line features and their surrounding features [54]. Straight line segments are extracted from the images and characterized by their position and orientation as well as their relationships with nearby straight

line segments, forming a *relational graph.* Feature matching is done by performing a graph search that finds mutually compatible nodes in this graph. Hellwich and Faig improve on the relational matching scheme presented by Horaud and Skordas by extending it to curved lines and avoiding the use of epipolar constraints [51] [52]. Properties of the line segments such as corner points, curvature, average edge strength etc. are used to define a cost function that is minimized during a search of the relational graph between the line segments. The matching procedure starts by the prediction of hypothesis, which first tries to match significant edges in the image. Further potential matches are found by the propagation of the predicted hypothesis based on the relational "edge neighbourhood graph" and finally solve ambiguities based on the number of matched segment pairs. Zilberstein uses a similar *relational matching* technique to match scanlines of a rectified stereo pair [106]. Either the greylevels or the resultant of the convolution of the image with the LoG operator for each scanline is transformed to a tree representation of the one-dimensional signal. This tree representation takes into account the geometric properties of the scanline signal. Areas of the scanlines are matched using a graph search that tests shape and parallax for matching greylevels and zero-crossings of the LoG output.

Mason and Wong use unique combinations of lines called *line triples* as features for image matching [75]. Line triples present a very sparse point field, and is therefore recommended only for image alignment. Line segments are transformed to $\phi(s)$ space and characterized as being either straight or curved. Line triples are formed by grouping together triplets of connected line segments and matched according to the geometric properties of the line triples.

One of the first research pairs to formulate the image matching solution as a combination between area based matching and feature based matching was Cochran and Medioni [17]. They use image pyramids that are resampled in epipolar geometry to perform a coarse-to-fine match. A dense disparity map is generated using area correlation, and this disparity map is then refined using matched edge elements. The ABM and FBM steps are performed in parallel and the results combined before proceeding to the next level in the image pyramid. The main disadvantage of this matching scheme is that it is computationally very expensive.

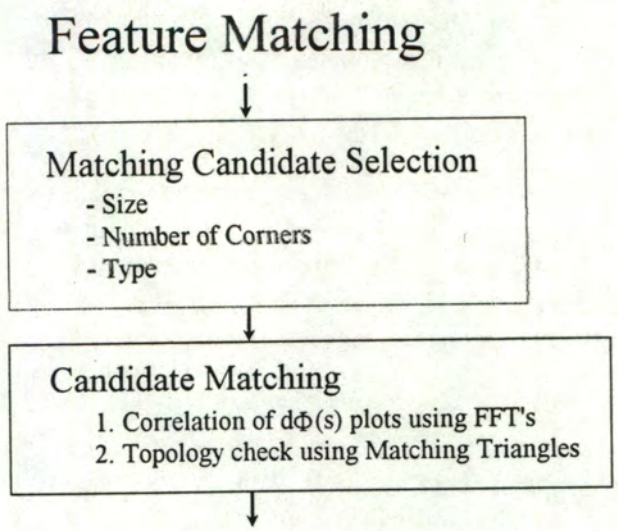


Figure 6.1: Feature Matching Flowchart

In this chapter the feature matching algorithm designed during this research project will be presented. First the method of identifying possible feature matching candidates using the feature vectors as described in chapter 5 will be described. Next the two phase technique to match the feature matching candidates will be described in detail. During the first phase, matching probabilities are found by calculating the correlation values between the feature descriptors of matching candidates. In the second phase the most probable matches are verified by observing feature topology. The whole matching process will be described through the use of an example used in this research project.

6.1 Feature Matching Using $d\phi(s)$ Plots and Matching Triangles

A distinct advantage of Feature Based Matching (FBM) schemes above other matching schemes is that the search space for matching points, lines and regions can dramatically be reduced. The information obtained during feature classification makes it possible to narrow down the search for matching features by comparing the relevant feature vectors described in chapter 5. Throughout this chapter it will be assumed that the subject image is the left image of the stereo-pair and that for each feature in the left image the matching feature in the right, or candidate image must be found.

In this thesis project possible matching candidates, called *Feature Matching Candidates* (FMC's) are selected by comparing the feature vectors *size*, *type* and *number of corners*. Only features of the same type and similar size and number of corners are accepted as FMC's. From these FMC's, which form only a subset of all the features in the candidate image, the correct matches are found using a two-phase matching process. During the first phase of this matching process an ordered Feature Correlation Table (FCT) is generated by calculating matching probabilities from the correlation functions between the feature descriptors of the FMC's. In this project the feature descriptor used for correlation is the $d\phi(s)$ plot, the first derivative of the $\phi(s)$ plot. During the second phase of the feature matching process the most probable feature matches as indicated by the entries in the Feature Correlation Table are verified by examining the topology between the relevant features. This is achieved by searching for matching triangles that are formed between the centre-of-mass points of adjacent features. The features forming the vertices of the *feature triangles* are determined by the entries in the FCT and the feature vectors.

The advantage of using this two-phase matching scheme is in strengthening possible matches and solving ambiguities by taking the feature topology into account. Where the correlation function between the feature descriptors takes only the shape of the feature into account, the topology of the feature describes the spatial relationship of that feature relative to its neighbours. This extra relational information was successfully implemented in line matching schemes described by Horaud and Skordas [54] and Hellwich and Faig [51] [52]. An obvious case where the feature topology is of use is to solve ambiguities due to the repetition of features. If one particular feature is repeated often the correlation functions for each of the repeated features will be high and will not necessarily be able to distinguish which feature is the correct match.

6.1.1 Selecting Candidates for Feature Matching

The candidates for a possible match are selected by comparing the feature vectors that describe the features. Only the features that have the same or similar feature vectors are considered as Feature Matching Candidates (FMC's), from which the correct matches are found using the two-phased matching process described later on in this chapter. In this project the *size*, *type* and *number of corners* are compared during this selection process. The following criteria for selecting matching candidates was used:

1. The feature *size* of the candidate must be within ten percent of the size of the subject feature

2. The *number of corners* must not differ by more than one

3. The feature *type* must be the same

The feature *sizes* are compared according to the Euclidean distance between the two vectors i.e. if the left feature has length N_1 and the right feature has length N_2 then the Euclidean distance as a percentage of the resultant length is defined as

$$d = 100\frac{|N_2 - N_1|}{\sqrt{N_1^2 + N_2^2}} \tag{6.1}$$

A difference of one *corner* has been allowed for to take into account the sensitivity of the threshold value in the corner detection step. As was shown the threshold can be just too high to mark a corner position which could be found in the corresponding image. Corners that are not visible in one image due to the viewing angle could become visible in another image from a different viewing angle. A classic example of this is with aerial images of a pitched rooftop. Viewed from directly above the top of the pitch can not be distinguished from the side of the building, but from a different viewing angle the top of the pitch is visible as a corner in the feature outline.

The feature *type* broadly classifies a feature into three basic types. The feature types that have been defined are ARCS, CIRCLES and POLYGONS. An ARC is a linked edge chain that has two nodes at the endpoints that don't meet at the same point i.e. an open line feature. A POLYGON is a closed line feature where the start and end nodes are at the same point. A CIRCLE is defined as a polygon that has zero corners and with the ratio between the mean and standard deviation of the $d\phi(s)$ plot below a certain level. This step in the selection process attempts to emulate the visual perception of a human by for example not trying to match a circle to a rectangle.

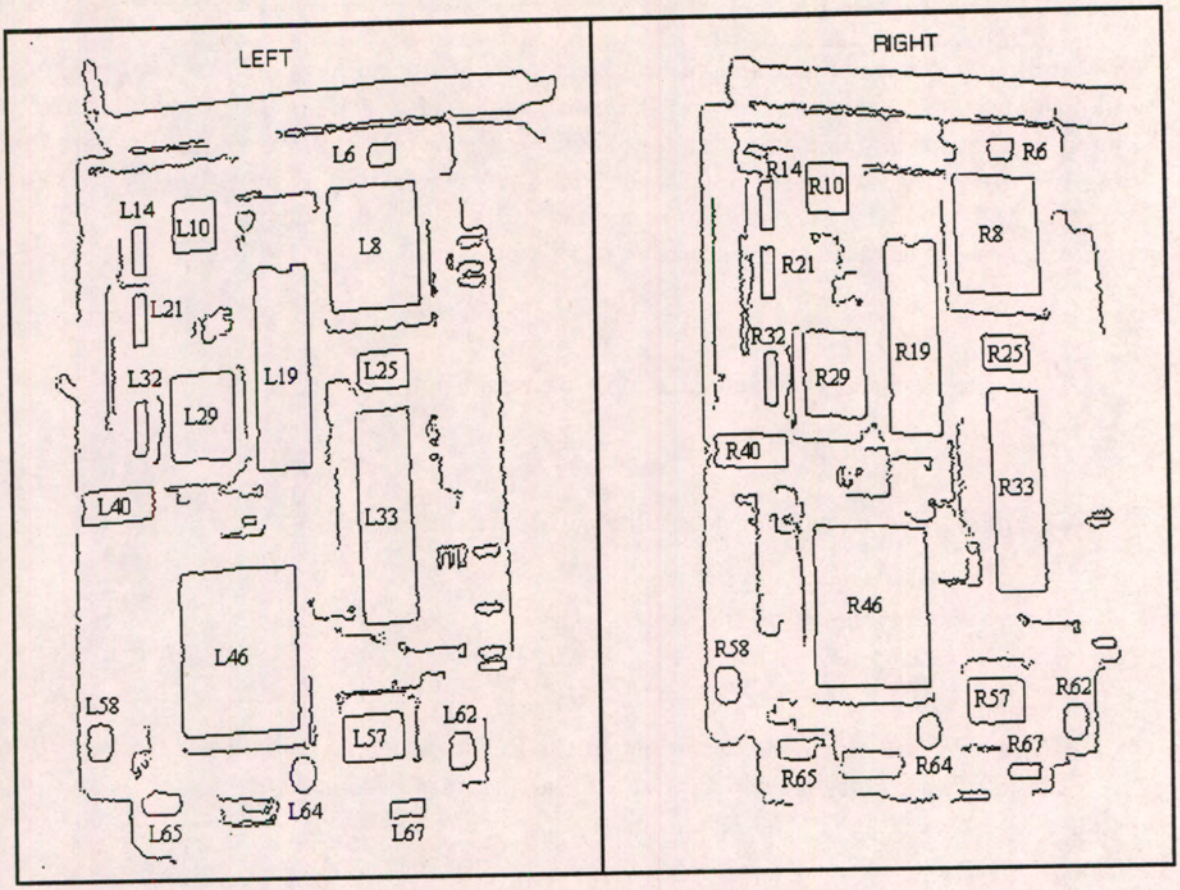


Figure 6.2: LEFT and RIGHT image features after extraction and classification

The reader is referred to figure 6.2 for a plot of the LEFT and RIGHT image features after feature extraction and classification. The image features are obtained from a stereo pair of PCB images.

The features have been classified and the results of the classification can be found in table 6.1. For ease of comparison the Feature Classification Table has been arranged so that matching LEFT and RIGHT images features appear next to each other with corresponding numbers for ease of interpretation. Throughout this chapter examples using LEFT features labelled L.. and RIGHT features labelled R.. will refer to the features as labelled in figure 6.2.

LEFT	Type	Size	Corners	RIGHT	Type	Size	Corners
L6	POLY	50	3	R6	POLY	48	4
L8	POLY	237	4	R8	POLY	221	4
L10	POLY	104	4	R10	POLY	102	4
L14	POLY	69	3	R14	POLY	69	4
L19	POLY	283	5	R19	POLY	268	6
L21	POLY	71	3	R21	POLY	73	3
L25	POLY	94	4	R25	POLY	87	4
L29	POLY	160	4	R29	POLY	157	4
L32	POLY	74	3	R32	POLY	71	3
L33	POLY	288	4	R33	POLY	273	4
L40	POLY	113	4	R40	POLY	115	4
L46	POLY	311	4	R46	POLY	303	4
L57	POLY	116	4	R57	POLY	105	4
L58	CIRCLE	60	0	R58	CIRCLE	61	0
L62	CIRCLE	61	0	R62	CIRCLE	56	0
L64	CIRCLE	53	0	R64	CIRCLE	51	0
L65	CIRCLE	56	0	R65	POLY	67	3
L67	POLY	56	4	R67	POLY	50	2

Table 6.1: Feature Classification Table for matching LEFT and RIGHT image features

Note that only polygons and circles have been used in this example. Arcs were very seldomly repeated in the right image and were thus not used in the feature matching scheme.

6.1.2 Matching $d\phi(s)$ plots Using Correlation

In the field of signal processing a measure that gives an indication of the similarity between two random signals is the *Correlation* function [81] [96]. One can investigate the correlation between different but similar data sets by comparing them when they are superimposed with a relative shift left or right to build up a "time"-dependent correlation function, with the "time" representing the relative shift between the data sets [82]. The $d\phi(s)$ plots between different images should be similar for the same feature, and the correlation function can be used to quantify this similarity. The $d\phi(s)$ plot is a one-dimensional function representing the two-dimensional shape of the feature, and the "time" variable used in this case is "s", the distance travelled along the feature boundary.

Correlation in the Time-Domain

The correlation function $\Phi_{gh}(t)$ between two continuous functions $g(t)$ and $h(t)$ is defined as:

$$\Phi_{gh}(t) = \int_{-\infty}^{\infty} g(\tau + t)h(\tau)d\tau \tag{6.2}$$

which is a function of the lag or shift t between g and h.

The correlation between a function and itself is called the *autocorrelation*, which is used to calculate the amount of power contained in the signal.

The correlation function for two discrete functions $g[n]$ and $h[n]$ of length N is defined in a similar fashion as:

$$\Phi_{gh}[n] = \sum_{m=0}^{N-1} g[n + m]h[m] \tag{6.3}$$

where n is the lag in samples between the two data sets.

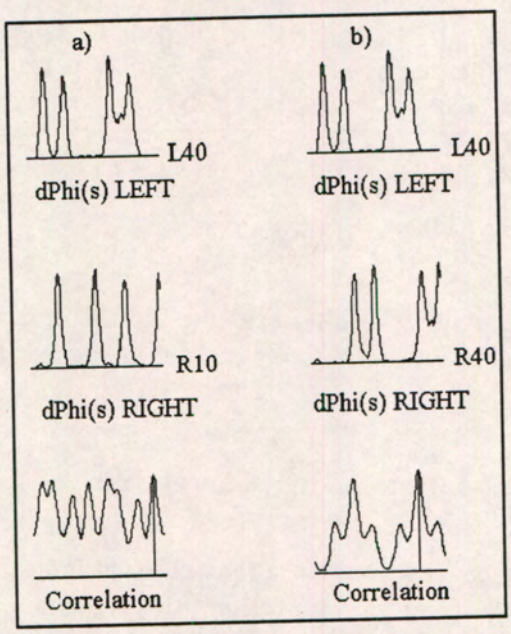


Figure 6.3: Time-domain correlation functions between LEFT 40 and RIGHT 10 a) and LEFT40 and RIGHT 40 b)

In figure 6.3 we see the $d\phi(s)$ plots for feature number L40 of the left image compared to that of features number R10 and R40 of the right image. Figure 6.3 a) shows the correlation function of feature R10 obtained through the time domain correlation function . The correlation function has many peaks of which the maximum correlation is 54.4 %. In figure 6.3 b) the correlation function for the correct candidate number R40 is shown. There is one local maxima peak that clearly shows the point of maximum correlation, in this case 95.3 %.

In section 5.1 on feature descriptors it was noted that the $d\phi(s)$ plot is invariant under rotation and translation, but the "s" or "time" axis is stretched and compressed due to the scale. To

take the scale difference between the left and right images into account the $d\phi(s)$ plots are resampled to the same length. Only one of the two functions being correlated needs to be resampled to the same length as the other to correct for the scale difference. Resampling can be done through bilinear interpolation, described in the appendix, or some other resampling method.

For signals of length N, the time-complexity of this method is calculated to be N^2 as follows:

Operation	Time-Complexity
N correlations per lag	N
N lags	N
Total	N^2

Table 6.2: Time-complexity for time-domain correlation

Correlation in the Frequency-Domain

In the frequency-domain, the Fourier Transform $\Phi_{gh}(\omega)$ of the correlation can be related to the Fourier Transforms $G(\omega)$ and $H(\omega)$ of the continuous signals $g(t)$ and $h(t)$ as:

$$\Phi_{gh}(\omega) = G(\omega)H^*(\omega) \tag{6.4}$$

where $H^*(\omega)$ is the complex conjugate of $H(\omega)$.

For real valued functions such as the "$d\phi(s)$" plots the imaginary part of the signal is zero and the complex conjugate is simply $H(-\omega)$.

The same result holds for real-valued discrete functions $g[n]$ and $h[n]$ with Fourier Transform pairs $G(\Omega)$ and $H(\Omega)$

$$\Phi_{gh}(\Omega) = G(\Omega)H(-\Omega) \tag{6.5}$$

where $\Phi_{gh}(\Omega)$ is the Fourier Transform of the discrete-time correlation function $\Phi_{gh}[n]$.

The discrete-time correlation function $\Phi_{gh}[n]$ can now be reconstructed by calculating the Inverse Fourier Transform of $\Phi_{gh}(\Omega)$

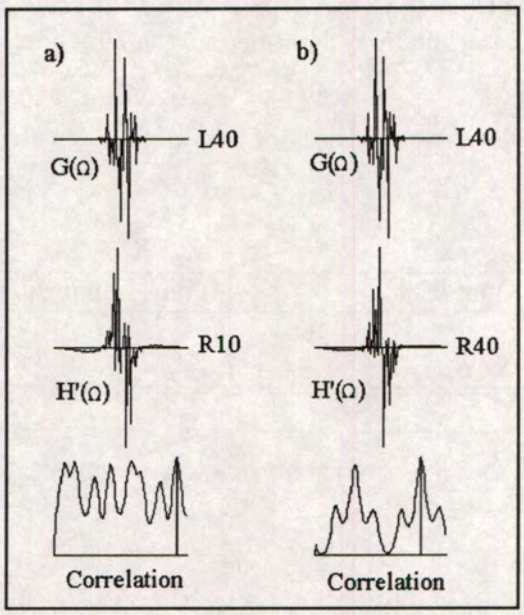


Figure 6.4: Frequency-domain correlation functions between features LEFT 40 and RIGHT 10 a) and LEFT 40 and RIGHT 40 b)

Figure 6.4 shows the correlation functions between features L40 and R10 in 6.4 a) and L40 and R40 in 6.4 b). The correlation functions are calculated using the Fourier Transforms of the $d\phi(s)$ plots, as expressed in equation 6.5. At the top of figure 6.4 a) the Fourier Transform $G(\Omega)$ of the $d\phi(s)$ plot for feature L40 is shown. This Fourier Transform is multiplied by $H(-\Omega)$, which is the complex conjugate of the function $H(\Omega)$, the Fourier Transform of the $d\phi(s)$ plot for feature R10. The resultant time-domain correlation function $\Phi_{gh}[n]$ is now found by calculating the Inverse FFT of the product function $G(\Omega)H(-\Omega)$. This resultant correlation function is shown at the bottom of figure 6.4 a).

Figure 6.4 b) shows the Fourier Transforms and resultant correlation function when comparing features L40 and R40. Note that in both cases a) and b) the resultant correlation functions are the same as when the correlation function is calculated using time-domain correlation. The resultant maximum correlation values are 54.4% and 95.3% respectively, exactly the same as for time-domain correlation.

As discussed in section 3.2 on Fourier Transforms, the FFT algorithm requires the input data set to be of an integral power of two in length. The next integral power of two above the feature size is calculated for each of the the left feature and the right candidate and the $d\phi(s)$ functions are resampled to these value. This resampling again takes into account the scale difference between the left and right images.

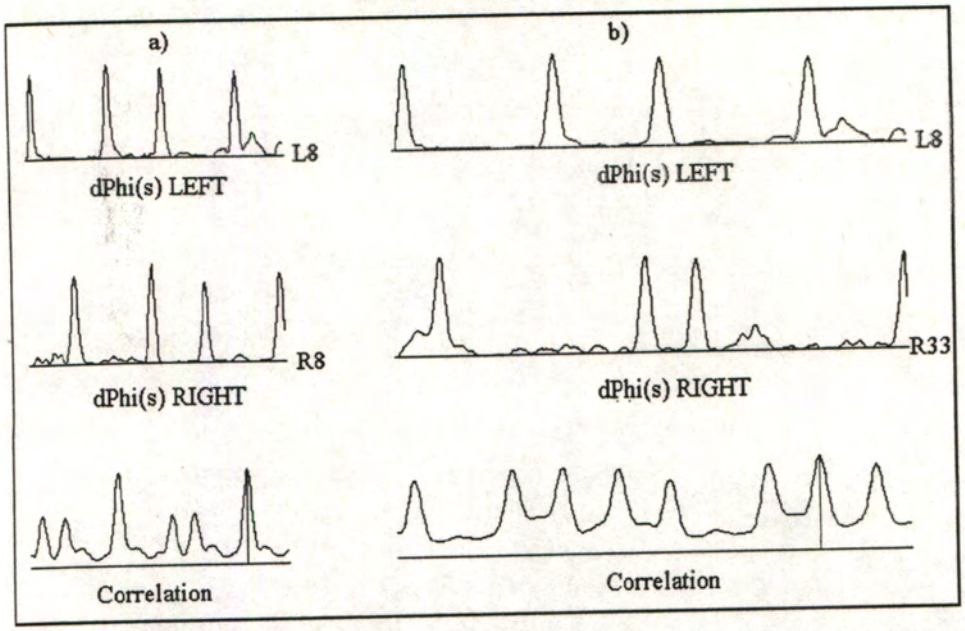


Figure 6.5: Different length correlation functions between LEFT 8 and RIGHT 8 a) and LEFT 8 and RIGHT 33 b)

In figure 6.5 an example is shown of a problem that arises in choosing the new length to which the $d\phi(s)$ plots should be resampled. The new length is chosen as the next integral power of two above the feature size. If the length of the LEFT feature is just less than an integral power of two then a possible matching RIGHT candidate could have a length greater than this value and still have Euclidean distance within 10 percent of the feature length. If the LEFT feature has to be resampled to N then the RIGHT candidate will have to be resampled to $2N$. A check for this situation arising has to be included to ensure that both plots are resampled to the same value. For ease of implementing the resampling algorithm the functions were always resampled to $2N$. Alternatively both the functions can be resampled to N which will be closer to the original feature sizes and computationally less expensive.

Figure 6.5 a) shows the $d\phi(s)$ plots for features L8 and R8 respectively. Feature L8 has length 237 and feature R8 has length 221 and both are resampled to the next integral power of two i.e. 256 . In figure 6.5 b) the right feature R33 is of length 273 and thus has to be resampled to 512 which is twice the length as for the first case. The correlation function between features L8 and its correct match R8 yields a maximum correlation of 95.2 % and the correlation function between L8 and R33 yields a maximum correlation of only 59.6 %.

Using the FFT algorithm to compute the correlation function [82], the time-complexity of this method for a sequence of length N is calculated to be $3Nlog_2N + 2N$ as follows:

Operation	Time-Complexity
2 FFT's	$2Nlog_2N$
1 Conjugate	N
1 Product	N
1 Inverse FFT	$Nlog_2N$
Total	$3Nlog_2N + 2N$

Table 6.3: Time-complexity for frequency-domain correlation using FFT's

From figure 6.6 it is immediately obvious that it is computationally more economical to calculate the correlation function using FFT's than it is to calculate it in the time-domain. For example the number of operations for a sequence of 64 values is 4096 in the time-domain compared to 1280 in the frequency domain. As the data sets get larger the differences in time-complexity become more prominent.

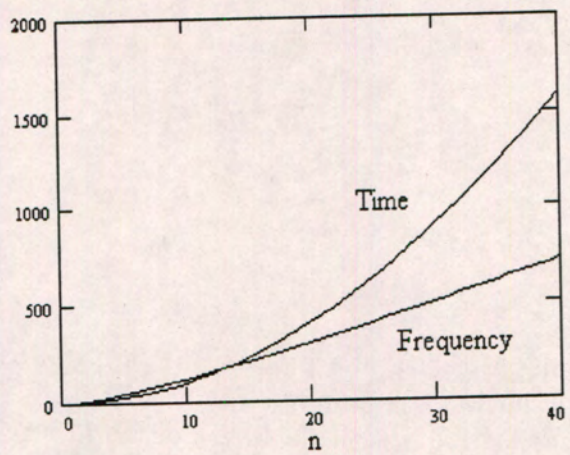


Figure 6.6: Time-complexity for the autocorrelation function in the time- and frequency domains

In this thesis project the correlation calculation between the relevant $d\phi(s)$ plots was performed using correlation in the frequency domain as described in this section. Although correlation in the time- and frequency domains yields the same final result, correlation in the frequency domain was preferred due to the better computational efficiency as depicted by figure 6.6.

The Correlation Coefficient

The output of the cross-correlation between signals $g[n]$ and $h[n]$ cannot be compared as is, as it is dependent on the width N of the sequence and the scale. To obtain a quantity that can be used for comparison we have to normalize the cross-correlation output. This normalized

quantity can directly be used for matching purposes and is called the *correlation coefficient.*

The correlation coefficient ρ is normalized by taking into account the autocorrelations of the signals $g[n]$ and $h[n]$.

$$\rho = \frac{(\Phi_{gh})_{max}}{\sqrt{\Phi_{gg}\Phi_{hh}}} \tag{6.6}$$

where Φ_{gh} is the crosscorrelation between $g[n]$ and $h[n]$, Φ_{gg} is the autocorrelation for $g[n]$ and Φ_{hh} is the autocorrelation for $h[n]$.

From equation 6.6 it can be deduced that for a perfect match where $g[n] = h[n]$ the correlation coefficient $\rho = 1$. The autocorrelation value Φ_{gg} is equal to $\Phi_{gg}[0]$, which is the maximum value of the correlation function $\Phi_{gg}[n]$. If $g[n] = h[n]$ then $\Phi_{gg} = \Phi_{hh}$ and both will be the same as $(\Phi_{gh})_{max}$.

If there is no correlation between $g[n]$ and $h[n]$ the correlation coefficient $\rho \rightarrow 0$.

6.1.3 Analysis of Feature Correlation

From the correlation coefficients between the LEFT features L6 to L67 and all potential matching RIGHT features a Feature Correlation Table is obtained which orders the possible matches according to the correlation coefficients. The correlation coefficients between left and right features as expressed as a percentage i.e. 100ρ where ρ is the correlation coefficient as defined in equation 6.6. In this specific example using the PCB images only the two most probable candidates are shown. The Feature Correlation Table for the numbered features of interest is shown in table 6.4.

Left	Right1	Correlation(%)	Right2	Correlation(%)
L6	R6	94.12	R67	88.41
L8	R8	95.23	R33	59.61
L10	R10	96.11	R57	93.72
L14	R14	99.41	R32	98.43
L19	R19	92.59	R33	83.38
L21	R32	98.98	R14	96.79
L25	R57	95.55	R25	93.86
L29	R29	90.01	-	-
L32	R32	86.72	R65	85.17
L33	R33	87.05	R46	58.53
L40	R40	95.35	R57	64.18
L46	R46	97.21	R33	55.25
L57	R57	96.18	R10	89.88
L58	R62	94.55	R58	93.25
L62	R62	95.63	R58	94.51
L64	R64	98.37	R62	93.83
L65	R62	91.14	R64	88.96
L67	-	-	-	-

Table 6.4: Ordered Feature Correlation Table for Feature Matching Candidates

This table indicates that for example the feature L6 has most likely matching candidate R6 with a "probability" of 94.12%, and second most likely matching candidate R67 with a "probability" of 88.41%. Throughout this section the examples used in the analysis of feature correlation will refer to table 6.4, the Feature Correlation Table.

The Feature Correlation Table 6.4 gives an indication of the uniqueness of a feature. The ideal situation would be if only one correlation value for the correct matching feature were found. This unique match was found only for feature L29. For all the other features except feature L67 at least two or more possible candidates were found. The correlation coefficient between the $d\phi(s)$ plots of any two features can be found, but only Feature Matching Candidates are considered. A further constraint that ensured that only legitimate entries were made in the Feature Correlation Table was that the resultant maximum correlation value had to greater than a threshold value, in this case $\rho = 0.5$.

For features L6 and L10 to L25 the correlation coefficient of the second candidate labelled *Right2* is relatively high which indicates the repetition of features. Clear examples of this are features L14, L21 and L32. All three these features are exactly the same on the PCB and it is expected that for each of these features high correlations will be found for all three the possible candidates labelled R14, R21 and R32 in the RIGHT image. For features L14 and L32 the correct candidates namely R14 and R32 were found at the highest correlation position labelled as *Right1*.

For the relative orientation, the orientation of the right image is found with respect to the left image. For this reason the left perspective centre can be defined as the origin of the relative coordinate system i.e.

$$\begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = 0$$

and $Q_1 = I$, the identity matrix i.e. $\omega_1 = \phi_1 = \kappa_1 = 0$. An arbitrary scale is fixed by setting $X_2 = 1$.

As previously defined, $Q_1 = R^{-1}(\omega_1, \phi_1, \kappa_1)$ and $Q_2 = R^{-1}(\omega_2, \phi_2, \kappa_2)$. The rotation matrix R is orthogonal and thus has the specific property that $R^{-1} = R^T$, i.e. $Q = R^T$. The co-planarity equation now becomes

$$\begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix}^T \left[R^T \begin{pmatrix} u_{2n} \\ v_{2n} \\ f_2 \end{pmatrix} \otimes \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix} \right] = 0 \quad \text{for} \quad n = 1...N \tag{7.11}$$

The coplanarity equation now represents a non-linear set of equations which can be solved in the *Least Squares* sense for the five unknowns ω_2, ϕ_2 and κ_2, contained in the rotation matrix R^T, and Y_2 and Z_2. If we define the coplanarity function 7.11 as $F(\omega_2, \phi_2, \kappa_2, Y_2, Z_2)$ then we can linearize this function using the Taylor series expansion and keeping only the first order terms. The function is differentiated with respect to the five unknowns $\omega_2, \phi_2, \kappa_2, Y_2$ and Z_2 and expands to

$$F = F^0 + \frac{\partial F}{\partial \omega_2}\triangle\omega + \frac{\partial F}{\partial \phi_2}\triangle\phi + \frac{\partial F}{\partial \kappa_2}\triangle\kappa + \frac{\partial F}{\partial Y_2}\triangle Y + \frac{\partial F}{\partial Z_2}\triangle Z \tag{7.12}$$

This can be rewritten as the linear system

$$\begin{pmatrix} a^t_{11} & a^t_{21} & a^t_{31} & a^t_{51} & a^t_{61} \\ \vdots & & & & \\ a^t_{1N} & a^t_{2N} & a^t_{3N} & a^t_{5N} & a^t_{6N} \end{pmatrix} \begin{pmatrix} \triangle\omega \\ \triangle\phi \\ \triangle\kappa \\ \triangle Y \\ \triangle Z \end{pmatrix} = -X_2 \begin{pmatrix} a^t_{41} \\ \vdots \\ a^t_{4N} \end{pmatrix} - Y^t_2 \begin{pmatrix} a^t_{51} \\ \vdots \\ a^t_{5N} \end{pmatrix} - Z^t_2 \begin{pmatrix} a^t_{61} \\ \vdots \\ a^t_{6N} \end{pmatrix} \tag{7.13}$$

where

$$a^t_{1n} = \frac{\partial F}{\partial \omega_2}$$

$$= \begin{pmatrix} X_2 \\ Y_2^t \\ Z_2^t \end{pmatrix}^T \left[\frac{\partial R^T}{\partial \omega}(\omega_2^t, \phi_2^t, \kappa_2^t) \begin{pmatrix} u_{2n} \\ v_{2n} \\ f_2 \end{pmatrix} \otimes \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix} \right]$$

$$a_{2n}^t = \frac{\partial F}{\partial \phi_2}$$

$$= \begin{pmatrix} X_2 \\ Y_2^t \\ Z_2^t \end{pmatrix}^T \left[\frac{\partial R^T}{\partial \phi}(\omega_2^t, \phi_2^t, \kappa_2^t) \begin{pmatrix} u_{2n} \\ v_{2n} \\ f_2 \end{pmatrix} \otimes \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix} \right]$$

$$a_{3n}^t = \frac{\partial F}{\partial \kappa_2}$$

$$= \begin{pmatrix} X_2 \\ Y_2^t \\ Z_2^t \end{pmatrix}^T \left[\frac{\partial R^T}{\partial \kappa}(\omega_2^t, \phi_2^t, \kappa_2^t) \begin{pmatrix} u_{2n} \\ v_{2n} \\ f_2 \end{pmatrix} \otimes \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix} \right]$$

$$\begin{pmatrix} a_{4n}^t \\ a_{5n}^t \\ a_{6n}^t \end{pmatrix} = \begin{pmatrix} \frac{\partial F}{\partial X_2} & \frac{\partial F}{\partial Y_2} & \frac{\partial F}{\partial Z_2} \end{pmatrix}^T$$

$$= \begin{pmatrix} 0 & \frac{\partial F}{\partial Y_2} & \frac{\partial F}{\partial Z_2} \end{pmatrix}^T$$

$$= R^T(\omega_2^t, \phi_2^t, \kappa_2^t) \begin{pmatrix} u_{2n} \\ v_{2n} \\ f_2 \end{pmatrix} \otimes \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix}$$

Note that as X_2 has been selected as a constant, the term $\frac{\partial F}{\partial X_2}$ is zero, but has been included for completeness.

This equation system is solved for in the Least-Squares sense, where the design matrix is

$$A = \begin{pmatrix} a_{11}^t & a_{21}^t & a_{31}^t & a_{51}^t & a_{61}^t \\ \vdots & & & & \\ a_{1N}^t & a_{2N}^t & a_{3N}^t & a_{5N}^t & a_{6N}^t \end{pmatrix} \tag{7.14}$$

and the vector of unknowns

$$X = \begin{pmatrix} \triangle\omega \\ \triangle\phi \\ \triangle\kappa \\ \triangle Y \\ \triangle Z \end{pmatrix} \tag{7.15}$$

To solve for this system provisional values for each of these unknowns are assumed and iterated until the solution converges to the final values.

At iteration t we have

$$\begin{pmatrix} X_2 \\ Y_2^t + \triangle Y \\ Z_2^t + \triangle Z \end{pmatrix}^T \left[R^T(\omega_2^t + \triangle\omega, \phi_2^t + \triangle\phi, \kappa_2^t + \triangle\kappa) \begin{pmatrix} u_{2n} \\ v_{2n} \\ f_2 \end{pmatrix} \otimes \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix} \right] = 0 \tag{7.16}$$

The least-squares solution to the X-vector is used to update the unknowns for the next iteration using:

$$\begin{pmatrix} \omega^{t+1} \\ \phi^{t+1} \\ \kappa^{t+1} \\ Y^{t+1} \\ Z^{t+1} \end{pmatrix} = \begin{pmatrix} \omega^t \\ \phi^t \\ \kappa^t \\ Y^t \\ Z^t \end{pmatrix} + \begin{pmatrix} \triangle\omega \\ \triangle\phi \\ \triangle\kappa \\ \triangle Y \\ \triangle Z \end{pmatrix} \tag{7.17}$$

See the appendix for the relevant rotation matrix derivatives.

7.2 Epipolar-line Equations

" The epipolar line constraint is the strongest constraint in image matching and should be used as soon as available. Specifically it is independent of the shape of the object" (Haralick and Shapiro) [47].

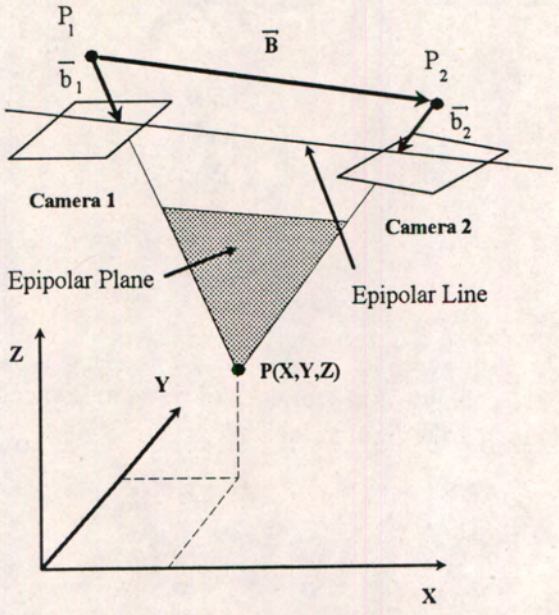


Figure 7.6: Epipolar Line Calculation

In a simple sense the *epipolar line* can be described as the locus of possible matching points in the candidate image for each individual point in the reference image. For each point in the reference image the matching point in the candidate image is found on a unique line, which reduces the two-dimensional image search to a one-dimensional search along the epipolar line. The epipolar line equation can be found using the interior and relative orientation information and solving for the co-planarity equation.

The vectors $\vec{B}$, $\vec{b_1}$ and $\vec{b_2}$ in figure 7.6 must all lie on the same plane, the *epipolar plane*. The epipolar line in the image is defined as the intersection between the epipolar plane and the image planes.

As above

$$\vec{B} = \begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix}, \quad \vec{b_1} = I \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix}, \quad \vec{b_2} = Q_2 \begin{pmatrix} u_{2n} \\ v_{2n} \\ f_2 \end{pmatrix}$$

These three vectors are all co-planar, thus $\vec{B} \cdot (\vec{b_1} \otimes \vec{b_2}) = 0$. During the relative orientation , the vector B between the perspective centres P_1 and P_2 and the rotation angles ω_2, ϕ_2 and κ_2 were found. The left rotation matrix was defined as the identity matrix I.

The unknown vector in equation 7.2 to be solved for is $\vec{b_2}$ for any specific point (u_{1n}, v_{1n}) in the reference image. The order of the vectors in the triple cross-product is irrelevant thus the coplanarity condition equation can be rewritten as $\vec{b_2} \cdot (\vec{B} \otimes \vec{b_1}) = 0$ i.e.

$$\begin{pmatrix} b_{2x} \\ b_{2y} \\ b_{2z} \end{pmatrix}^T \left[\begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} \otimes \begin{pmatrix} u_{1n} \\ v_{1n} \\ f_1 \end{pmatrix} \right] = 0 \tag{7.18}$$

This reduces to

$$b_{2x}(B_y f_1 - B_z v_{1n}) + b_{2y}(B_z u_{1n} - B_x f_1) + b_{2z}(B_x v_{1n} - B_y u_{1n}) = 0 \tag{7.19}$$

If the inverted rotation matrix is

$$Q_2 = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \tag{7.20}$$

then

$$b_{2x} = (m_{11}u_{2n} + m_{12}v_{2n} + m_{13}f_2) \tag{7.21}$$
$$b_{2y} = (m_{21}u_{2n} + m_{22}v_{2n} + m_{23}f_2) \tag{7.22}$$
$$b_{2z} = (m_{31}u_{2n} + m_{32}v_{2n} + m_{33}f_2) \tag{7.23}$$

Letting

$$a_1 = (B_y f_1 - B_z v_{1n}) \tag{7.24}$$
$$a_2 = (B_z u_{1n} - B_x f_1) \tag{7.25}$$
$$a_3 = (B_x v_{1n} - B_y u_{1n}) \tag{7.26}$$

Substituting for these values in equation 7.19, we get

$$(m_{11}u_{2n} + m_{12}v_{2n} + m_{13}f_2)a_1 + \tag{7.27}$$
$$(m_{21}u_{2n} + m_{22}v_{2n} + m_{23}f_2)a_2 + \tag{7.28}$$
$$(m_{31}u_{2n} + m_{32}v_{2n} + m_{33}f_2)a_3 = 0 \tag{7.29}$$

Rearranging and grouping we finally get the epipolar line equation for a particular left image point.

$$0 = M_1 u_{2n} + M_2 v_{2n} + M_3 f_2 \tag{7.30}$$

$$v_{2n} = -\frac{M_1}{M_2} u_{2n} - \frac{M_3}{M_2} \tag{7.31}$$

where

$$M_1 = (m_{11}a_1 + m_{21}a_2 + m_{31}a_3) \tag{7.32}$$
$$M_2 = (m_{12}a_1 + m_{22}a_2 + m_{32}a_3) \tag{7.33}$$
$$M_3 = (m_{13}a_1 + m_{23}a_2 + m_{33}a_3) \tag{7.34}$$

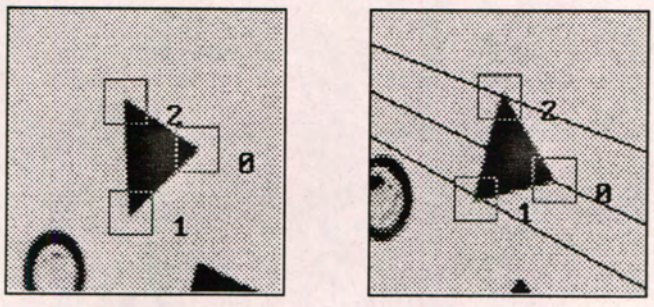


Figure 7.7: Epipolar Lines for the Corners of a Triangle

Figure 7.7 shows an example of the epipolar lines for the corners of a triangle. The corners in the reference sub-image are labelled 0, 1 and 2, with the corresponding epipolar lines as shown in the candidate image on the right hand side. In this example no lens distortion parameters were taken into account and the epipolar lines are straight lines, which is not the case when lens distortion parameters are taken into account.

7.3 Gross Error Detection

With any automated matching system the possibility of a mismatch must always be considered. The feature matching stage could generate incorrect matching pairs of features. In such a case the centre-of-mass point of an incorrect match will be an outlier that will affect the relative orientation calculation. In this thesis project error detection is done immediately after the relative orientation calculation.

A simple method of checking for an outlier resulting from the feature matching stage is to check the x and y parallax values for the matched features. This implies that the maximum depth change of the scene and the corresponding parallax values due to this depth change should be known *a priori*, which cannot be assumed in this case.

A more critical method of gross error detection is performed by checking the residuals of the relative orientation calculation. The relative orientation is thus performed first and then the outlier is detected using the "three sigma" test. In this case the residuals for each observation are checked against the standard deviation of the observation after the adjustment.

If the variance covariance matrix is Q_{ff} and the variance is σ_0^2 then the standard deviation σ_i of observation i after the adjustment is

$$\sigma_i = \sigma_0 \sqrt{Q_{ff_i}} \tag{7.35}$$

where Q_{ff_i} is the i^{th} diagonal element of the variance covariance matrix.

From this an outlier is detected if the absolute value of the residual v_i of observation i is larger than $3\sigma_i$ i.e.

$$|v_i| > |3\sigma_i| \tag{7.36}$$

would indicate an outlier.

Chapter 8

Area Based Matching

Line features that have been extracted and matched in the stereo pair represent the coarse structure of the image. According to Marr and Hildreth [74], these line features form part of a "primitive but rich" description of an image, called the *primal sketch.* With the matched line features as the "skeleton", the fine details can be "filled in" by employing area based matching techniques to match point features such as corners followed by a point-for-point matching of a subset of the matched line features.

The most widely used area based matching technique in digital photogrammetry is the Least Squares Matching (LSM) technique. In order for this technique to be implemented successfully the initial approximation for a matching point must be given to within 2-3 pixels of the correct position (*Baltsavias*) [7], which is a major disadvantage of this matching technique. If the correct starting position is not available the Least Squares Matching technique will fail to find the correct match.

The Least Squares Matching (LSM) process as described by Gruen *et al* [38] [40] [41] [49] [1] has been implemented successfully in many aerial and close-range photogrammetric systems. LSM refines a pixel-resolution point match to a sub-pixel match using a Least Squares fit. From an initial approximation for the matching point the LSM process attempts to match corresponding image patches using an affine transform to model the geometric distortion of the reference image patch. The corresponding image patches are updated and matched using a least squares model.

In this chapter the basic elements of the area based matching procedure will be described. The method of Least Squares Matching will be discussed first, followed by a discussion of obtaining initial matching positions using epipolar geometry. The calculation of the 3-D object coordinates from the collinearity equations using high-accuracy matching points is discussed. The chapter concludes with an explanation of the integration of epipolar geometry and the 3-D object coordinates into the matching process to add geometric constraints to the Least Squares Matching algorithm.

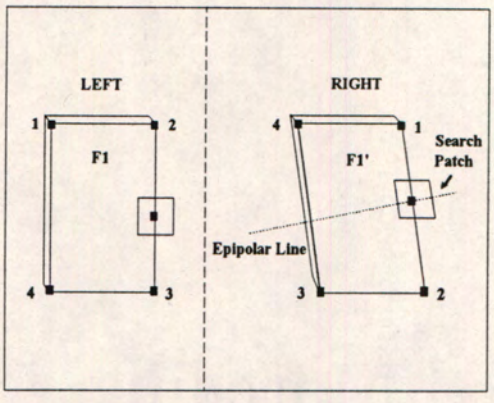


Figure 8.1: Initial Matching Point Approximations Using Epipolar Geometry

In this thesis project the initial approximations for matching points are obtained by using the epipolar geometry in conjunction with area correlation where necessary. The initial approximation for the matching point on a line feature in the reference image is obtained by finding the intersection between the corresponding line feature in the candidate image and the epipolar line for that particular point. If more than one intersection is found the ambiguity is resolved using area correlation. Finally, the initial approximation to the matching point is refined to sub-pixel level using the Least Squares Matching (LSM) technique.

8.1 Least Squares Matching

The Least Squares Matching process attempts to match the grey levels from a candidate image patch $g(x,y)$ to a reference image patch $f(x,y)$. The matching process thus tries to attain a perfect match such that:

$$f(x,y) = g(x,y) \tag{8.1}$$

In reality a perfect match will not be found due to image noise, occlusion, different radiometric factors due to different view angles for the two patches, to name but a few. We therefore introduce an error vector $e(x,y)$ such that

$$f(x,y) - e(x,y) = g(x,y) \tag{8.2}$$

gives a true representation of the system.

The image patches $f(x,y)$ and $g(x,y)$ each form a surface in three-dimensional space. This surface can be visualized as the x and y coordinates of the image patch, with the z-dimension the pixel grey levels, as shown in figure 8.2.

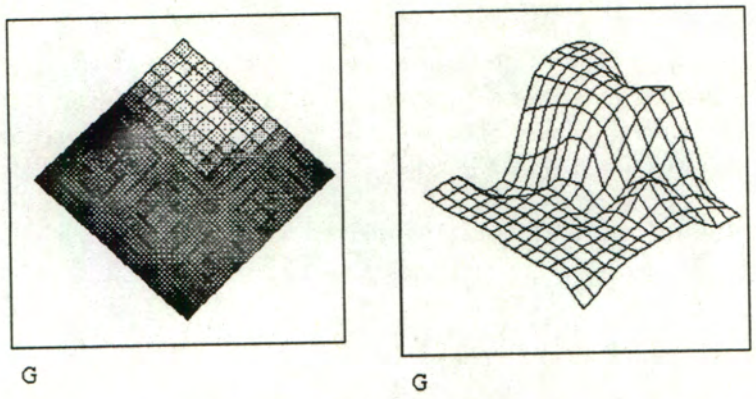


Figure 8.2: Grey scale image patch and 3-d surface visualization

The Least Squares model is based on the system-equation 8.2. This equation is a non-linear observation equation, and is linearized using the Taylor expansion, keeping only the zero- and first order terms. The "observations" in this equation system are the greylevels of the image patches $f(x,y)$ and $g(x,y)$ for each point (x,y) in the image patch.

$$f(x,y) - e(x,y) = g^0(x,y) + (\frac{\partial g(x,y)}{\partial x})dx + (\frac{\partial g(x,y)}{\partial y})dy \tag{8.3}$$

The image x- and y- gradients will be referred to as g_x and g_y respectively

$$\frac{\partial g(x,y)}{\partial x} = g_x \quad \text{and} \quad \frac{\partial g(x,y)}{\partial y} = g_y \tag{8.4}$$

The g_x and g_y gradients are calculated from the image patch using standard gradient based edge enhancement techniques such as greyscale differences, used by Baltsavias [7], or the Gradient-sum and Canny edge operators discussed in section 4.1 on edge detection.

The location and shaping of the candidate patch $g(x,y)$ is modelled according to the Affine Transformation model.

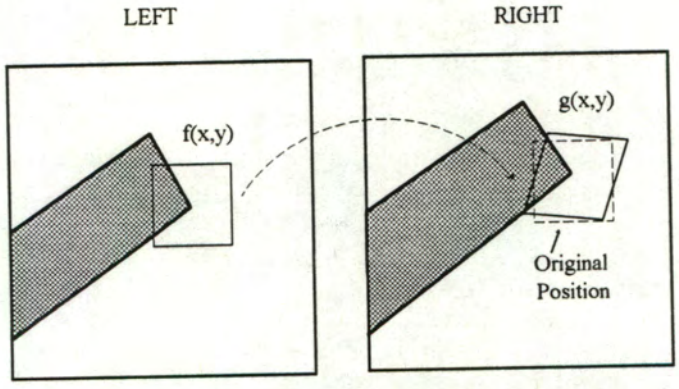


Figure 8.3: Least Squares Matching Process

The Affine coordinate transformation is a six parameter planar transformation that takes into account two translations, x-scale , y-scale and shears in x and y respectively. The shear factors result from a combination of a common rotation of the the orthogonal x and y axes followed by separate rotations of each axis.

Mathematically the transformation can be defined as

$$\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} u_{11} \\ v_{11} \end{pmatrix} + \begin{pmatrix} u_{12} & u_{21} \\ v_{12} & v_{21} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{8.5}$$

See the appendix for the formulation of the affine transformation model.

The position of each element $(\bar{x}, \bar{y})$ in the candidate patch $g(x, y)$ can be expressed in terms of the affine model as

$$\bar{x} = u_{11} + u_{12}x + u_{21}y \tag{8.6}$$

$$\bar{y} = v_{11} + v_{12}x + v_{21}y \tag{8.7}$$

where x and y are the coordinates of the original point position on the candidate patch $g(x, y)$.

The first order differentials are

$$d\bar{x} = du_{11} + du_{12}x + du_{21}y \tag{8.8}$$

$$d\bar{y} = dv_{11} + dv_{12}x + dv_{21}y \tag{8.9}$$

The Least Squares observation equation thus becomes

$$f(x,y) - e(x,y) = g^0(x,y) + g_x du_{11} + g_x x du_{12} + g_x y du_{21} + g_y dv_{11} + g_y x dv_{12} + g_y y dv_{21} \tag{8.10}$$

This represents the observation equation for the Parametric Case of Adjustment. The characteristics of the system can be summarized as follows:

- One observation equation is formulated for each observation
- Each observation equation contains one observation and one or more unknowns

The format of an observation equation in vector form is

$$\vec{v} = A\vec{X} - \vec{l} \tag{8.11}$$

where

$\vec{v}$ = vector of corrections or *residuals*
A = matrix of design coefficients or *design matrix*
$\vec{X}$ = vector of *unknown* parameters
$\vec{l}$ = vector of *free* terms

The vector of corrections is equivalent to the error vector e(x,y) , and the Least Squares principle minimizes the sum of the squared residuals. The sum is computed over the whole image patch, with each element in the patch being treated as an observation. In vector form this can be expressed as

$$v^T P v \Rightarrow Min \tag{8.12}$$

where P is the weight-matrix.

For this greyscale matching process the vectors at observation n, where n is incremented sequentially from the top-left pixel of the image patch, are as follows

$$X_n{}^T = (du_{11} \quad du_{12} \quad du_{21} \quad dv_{11} \quad dv_{12} \quad dv_{21}) \tag{8.13}$$

$$l_n = f(x,y) - g^0(x,y) \tag{8.14}$$

$$A_n = (g_{x_n} \quad g_{x_n}x \quad g_{x_n}y \quad g_{y_n} \quad g_{y_n}x \quad g_{y_n}y) \tag{8.15}$$

where A_n is a row of the A-matrix for observation n.

The solution vector can be expressed using matrix algebra as

$$\vec{X} = (A^T P A)^{-1} A^T P l \tag{8.16}$$

where the weight matrix P is a principal diagonal matrix, each diagonal element of which assigns a weight to that particular observation. For greyscale matching all the weights are assumed to be equal to unity, therefore $P = I$, the identity matrix. We now have

$$\vec{X} = (A^T A)^{-1} A^T l \tag{8.17}$$

This solution vector gives the corrections that have to be applied to the provisional values of the affine parameters. After updating the affine parameters with the corrections, new x and y parameters for each observation in $g(x,y)$ are calculated using the updated affine

model. These new positions will fall in between pixels of the candidate search window, and the greyscale values must be resampled at these positions using bilinear interpolation. See Appendix .1 for resampling using bilinear interpolation.

The process of recalculating the greyvalues in the candidate patch $g(x,y)$ by updating the affine model and resampling the candidate image patch is iterated until the corrections to the affine transformation in the Least Squares model converge to values below predetermined minimum threshold values or a certain number of iterations have been performed. If the maximum number of iterations has been performed before convergence then the correct matching position is not found.

In this thesis project only the x and y shifts are checked for convergence. If the resultant shift $d = \sqrt{du_{11}^2 + dv_{11}^2}$ becomes less than a preset threshold then the iteration stops and the matching point has been found.

Figure 8.4 shows an example of a successful match for a corner point in the stereo pair of a Printed Circuit Board (PCB). The left and right image windows of 64x64 pixels are shown, with the location of the image patches $f(x,y)$ in the left image window shown. The positions of $g(x,y)$ at the first and last iterations of the Least Squares matching scheme are also indicated, with the corresponding 15x15 image patches $f(x,y)$ and $g(x,y)$ shown under the relevant image windows.

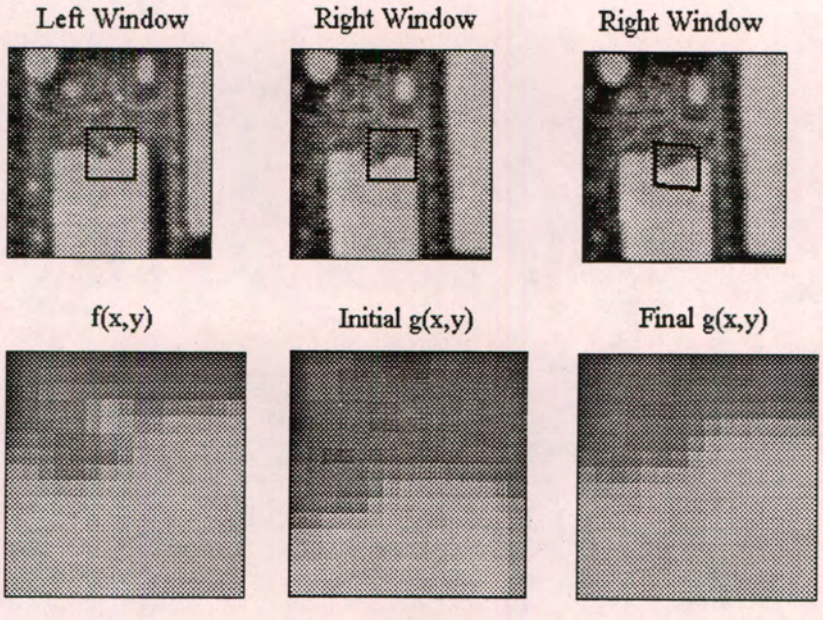


Figure 8.4: Successful Least Squares Match for a Corner Point

Figure 8.5 shows the affine correction parameters $du_{11}..dv_{21}$ as a function of the iteration number n. The x- and y-shift parameters du_{11} and dv_{11} are checked to determine if the Least Squares matching has converged. In this case the resultant shift d converged to less than 0.001 of a pixel after nine iterations. The rate of change of the shaping parameters affecting scale and shear has been limited, therefore these parameters had not converged completely by the time the x- and y-shift parameters had converged.

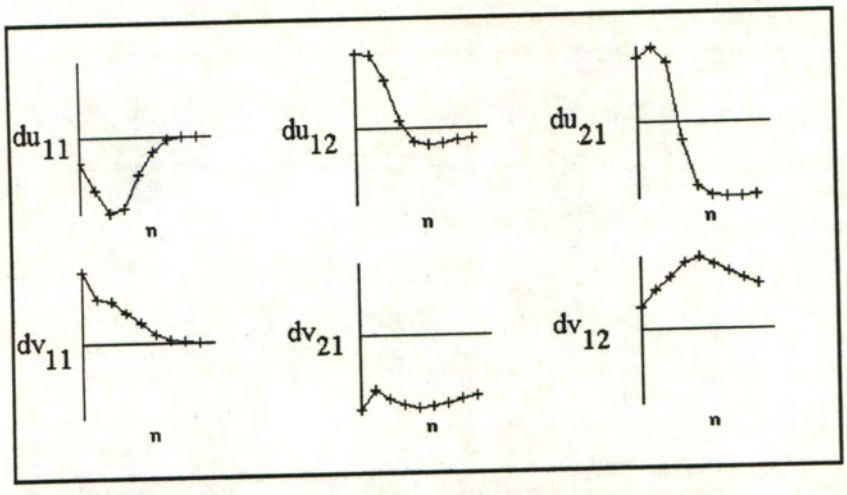


Figure 8.5: Affine Correction Parameters as a Function of the Iteration Number n

Figure 8.6 shows an example of a corner point from a PCB stereo pair that does not converge after twenty iterations of the Least Squares matching process. The relevant image initial and final image windows with corresponding 15x15 image patches $f(x,y)$ and $g(x,y)$ are shown.

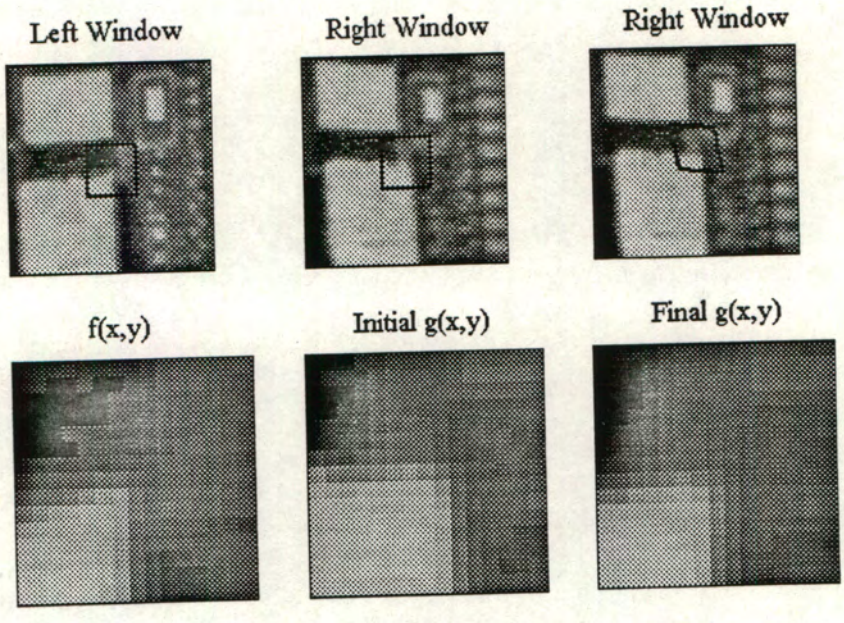


Figure 8.6: Unsuccessful Least Squares Match for a Corner Point

The reader is referred to figure 8.7 for plots of the affine correction parameters $du_{11}..dv_{21}$ as a function of the iteration number n. The y-shift parameter dv_{11} converges to zero, but the x-shift parameter du_{11} does not converge to a value small enough to satisfy the convergence criterion that the resultant shift d must be below 0.001 of a pixel. The parameter du_{12} converges to zero, but the parameter dv_{21} diverges, resulting in the convergence criterion never being met.

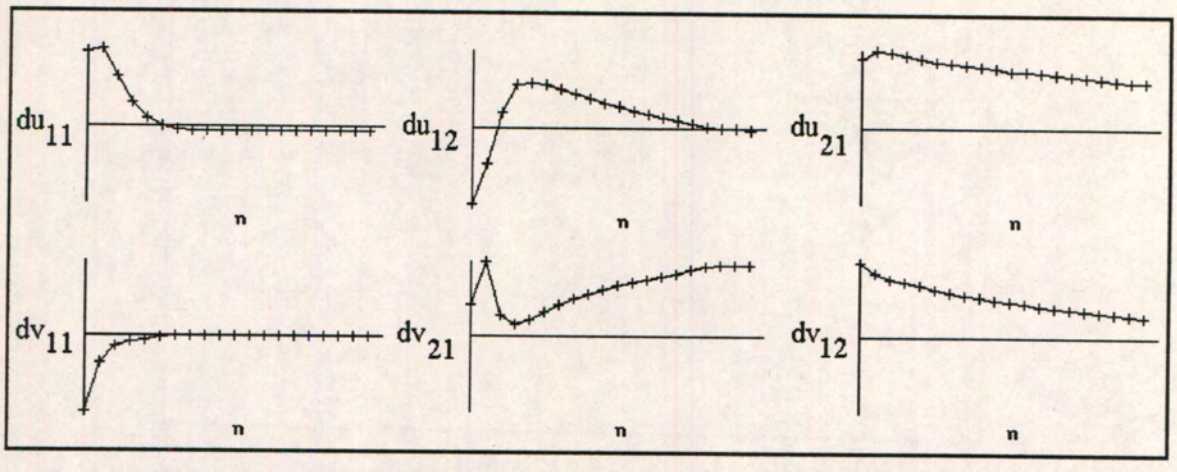


Figure 8.7: Affine Correction Parameters as a Function of the Iteration Number n Using the Least Squares Principle

To deal with corresponding image patches that don't converge using the Least Squares approach a different approach must be taken. Calitz and Rüther [14] report on using L-1 Norm methods for image matching. The method of *robust estimation* used in photogrammetric applications is described by Kubik *et al* [66] and can mathematically be defined by

$$\sum |v|^p \Rightarrow Min \qquad 1 \leq p < 2 \tag{8.18}$$

where $|v|$ is the absolute value of the residuals. Typical values for p is between 1 and 1.5. For $p = 1$ the matching principle is called the *Least Sum* principle or L1-norm. The Least Squares principle would require $p = 2$, which is referred to as the L2-norm.

The solution vector to the corrections in the affine model can be expressed using matrix algebra as

$$\vec{X} = (A^T P A)^{-1} A^T P l \tag{8.19}$$

During the Least Squares matching process the weights p_i for each observation i were unity, resulting in $P = I$, the identity matrix. For the Least Sum method the weights for each observation are indirectly proportional to the residual v_i

$$p_i = \frac{1}{|v_i| + \epsilon} \tag{8.20}$$

where ϵ is a small constant to avoid a singularity when $v_i = 0$.

Generally the Least Sum method reduces the negative effect of outliers and converges faster than the Least Squares method.

In this thesis project the Least Sum method was applied to the image patches if the Least Squares matching process did not converge after twenty iterations. Only if neither the Least Squares nor the Least Sum method converged was a corresponding pair deemed to be undefined.

Refer to figure 8.8 for the plot of the corrections to the affine parameters for the same example as shown in figure 8.6, but with the Least Sum matching principle applied. Note that all the affine parameters have converged after fourteen iterations, with the x-shift parameter du_{11} showing some oscillations before converging on the final value.

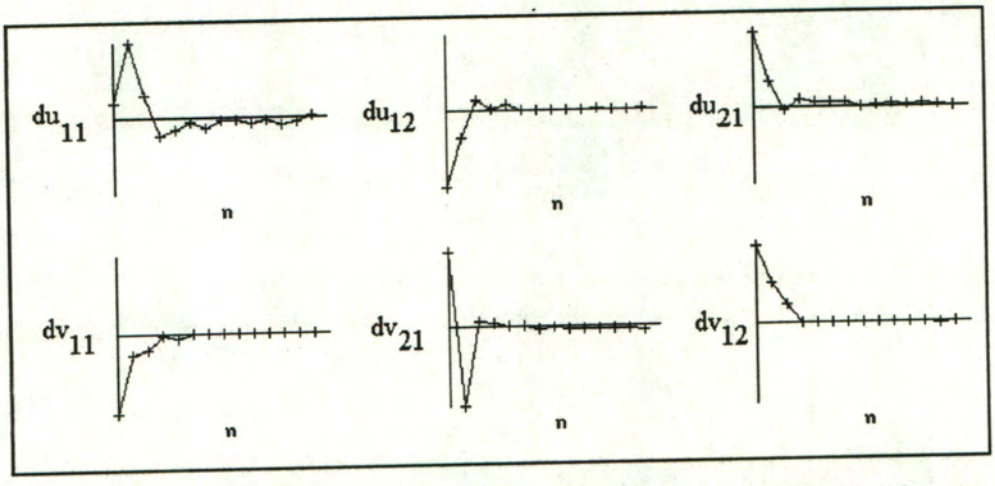


Figure 8.8: Affine Correction Parameters as a Function of the Iteration Number n using the Least Sum Principle

Weighting each observation by the inverse of the residual for that observation dampens the nonlinear equation system modelling the matching process. Damping a nonlinear system effectively forces the response of a system to quickly converge to a final value by limiting the size of the oscillations. The effect of each parameter in the Least Squares model can also be damped by adding large weights to the diagonal entries in the A^TPA matrix corresponding to that parameter. In this example weights of one million were used.

In this thesis project all the variables except for the x- and y-shifts were weighted to limit the rate of change of the shaping parameters affecting scale and shear.

Figure 8.9 shows an example of a successful match found for a corner point in a PCB image with all the shaping parameters remaining free i.e. no weights have been added to any of the diagonal elements of the A^TPA matrix.

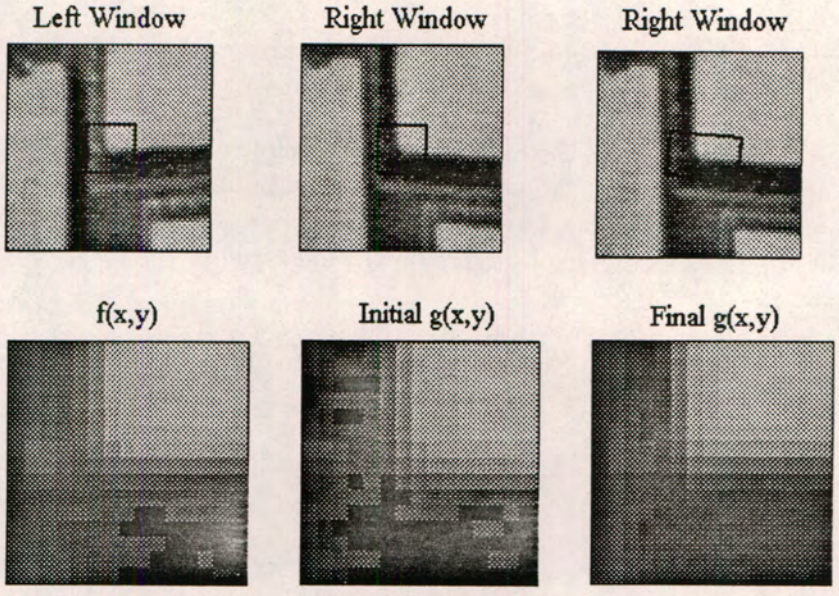


Figure 8.9: Successful Least Squares Match With All Shaping Parameters Free

From figure 8.10 it can be seen that all six the shaping parameters converge as the corrections to the shaping parameters shown all become insignificant after eleven iterations.

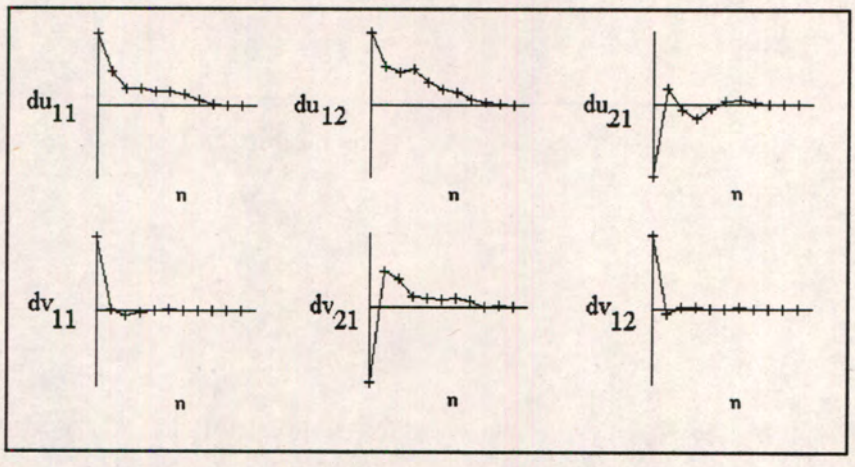


Figure 8.10: Convergent Shaping Parameter Corrections With All Shaping Parameters Free

Generally the Least Squares matching with all shaping parameters free converged faster than when weights were added to constrain the scale and shear shaping parameters in cases of high correlation between f and g. If however there is a relatively poor correlation between f and g the matching system can become unstable and oscillate or diverge. Figure 8.11 shows the shaping parameter corrections for the same example as used in figure 8.4 but with all the shaping parameters kept free. Note that all the parameters except the y-shift become unstable and either oscillate or diverge. With weights added to constrain the system the x- and y-shifts converged after nine iterations, but without constraints the solution does not converge in either L1 or L2.

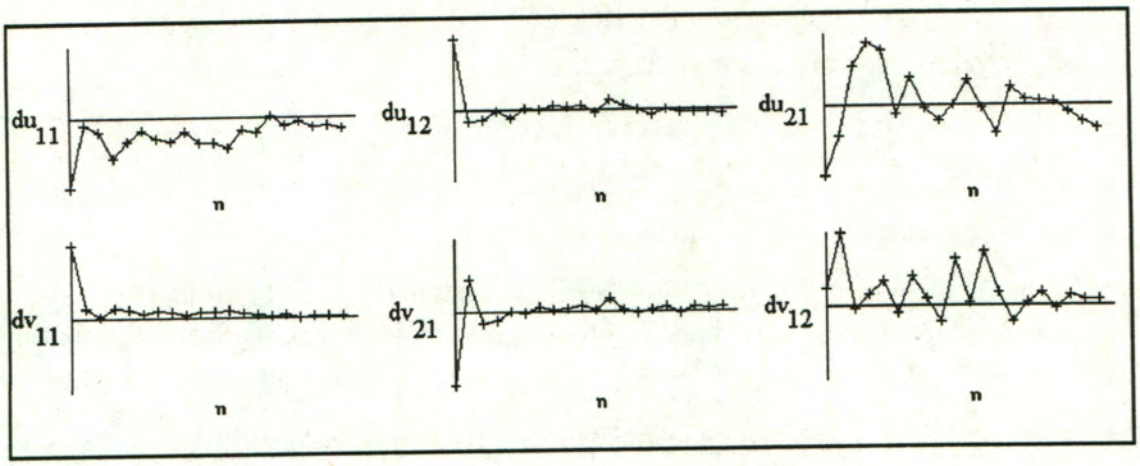


Figure 8.11: Divergent Shaping Parameter Corrections With All Shaping Parameters Free

In the example of the PCB stereo pair there are 45 corner points that need to be matched using the Least Squares or Least Sum matching scheme. With all the shaping parameters free all but two matches converge, with a total number of iterations of 514. If only the x- and y-shift parameters are kept free then the matching scheme performs marginally better by converging on all but one occasion, but with a larger total number of iterations of 589.

Refer to figure 8.12 for the matching corner points obtained from Least Squares matching with constraints on the shaping parameters. Corners matched using Least Squares Matching are indicated by the crosses and the matched feature centre-of-mass points are shown as rectangles and circles. These matching corner points are used to update the relative orientation calculation.

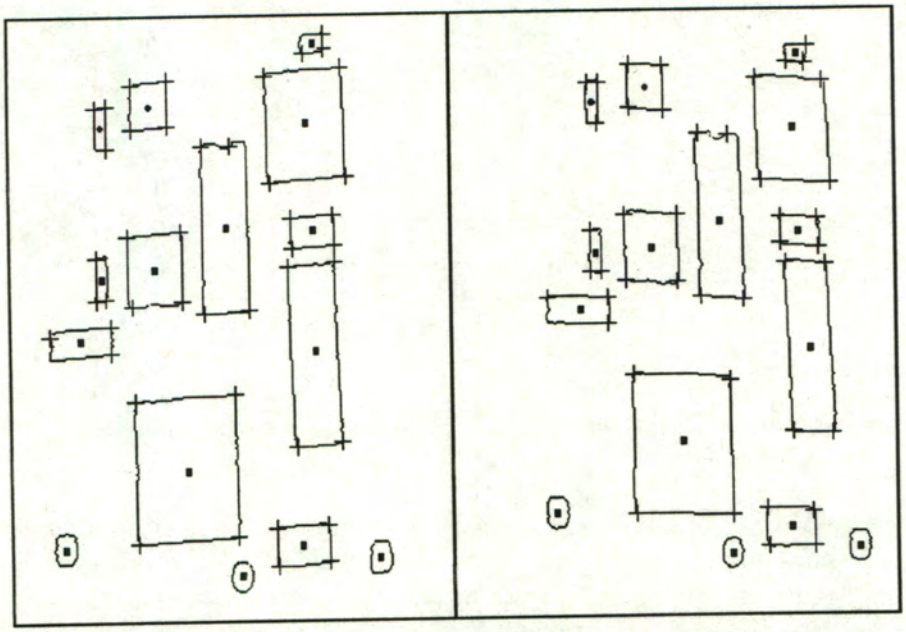


Figure 8.12: Matching Corner Points for Updating the Relative Orientation

8.2 Radiometric Corrections

In greyscale matching, allowances must be made for intensity changes due to image noise, varying spectral properties, occlusion and intensity changes due to different viewing angles to name but a few.

The intensity differences due to the viewing angle can be explained in terms of the model for the reflected light incident on a point in object space. For a perfect lambertian radiator, the intensity of the light reflected back is given by the cosine function i.e.

$$I = I_o cos(\theta) \tag{8.21}$$

where I_o is the normal reflected intensity and θ is the viewing angle relative to the normal.

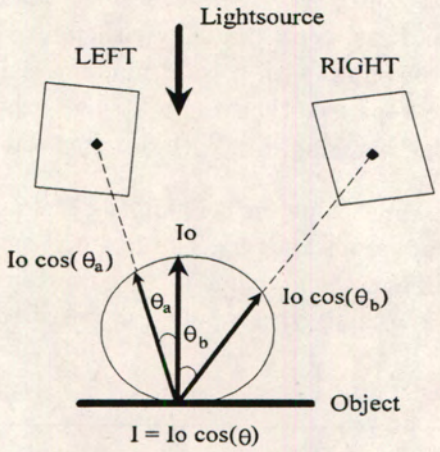


Figure 8.13: Intensity difference due to a Lambertian radiator

Generally the amount of backscatter can be modelled as a cosine of order n

$$I = I_o cos^n(\theta) \tag{8.22}$$

From this it is clear that the same point will have different intensities when viewed from different angles. For this reason radiometric corrections are applied to try and restore the balance.

The difference in viewing angle is only one of many factors resulting in different image intensities for the same object point. The most common radiometric transformation between the reference image patch $f(x, y)$ and the candidate image patch $g(x, y)$ is a linear transformation (Haralick and Shapiro) [47] (Baltsavias) [7].

Haralick and Shapiro model the area based matching system as

$$h_1(x,y) = f(x,y) + n_1(x,y) \tag{8.23}$$
$$h_2(x,y) = G[f(x,y)] + n_2(x,y) \tag{8.24}$$

where n_1 and n_2 are due to image noise, which is incorporated in the error vector $e(x,y)$ introduced in equation 8.2. The radiometric transformation $G[f(x,y)]$ represents a linear transformation between $f(x,y)$ and $g(x,y)$ i.e.

$$f(x,y) = r_1 g(x,y) + r_0 \tag{8.25}$$

where the constant r_1 is the *gain* factor and r_0 represents a *shift* in the dynamic range of the greylevels. The dynamic range of the greylevels refers to the range h_{min} to h_{max} of greylevels appearing in an image patch $h(x,y)$.

The Wallis filter was used in this thesis project to transform the greylevels of the candidate patch using a linear transformation

$$g'(x,y) = [g(x,y) - \mu_g]\frac{\sigma_f}{\sigma_g} + \mu_f \tag{8.26}$$

where μ_g and σ_g are the mean and standard deviation of the candidate image patch $g(x,y)$ and μ_f and σ_f are the mean and standard deviation of the reference image patch $f(x,y)$.

The mean μ_h and standard deviation σ_h for any image patch $h(x,y)$ can be calculated as

$$\mu_h = \frac{1}{(2J+1)(2K+1)} \sum_{x_1=-J}^{J} \sum_{y_1=-K}^{K} h(x+x_1, y+y_1) \tag{8.27}$$

and

$$\sigma_h^2 = \frac{1}{(2J+1)(2K+1)-1} \sum_{x_1=-J}^{J} \sum_{y_1=-K}^{K} [h(x+x_1, y+y_1) - \mu_h]^2 \tag{8.28}$$

where $2J+1$ and $2K+1$ are the x and y dimensions of the image patch $h(x,y)$.

Note that the Wallis filter shown in equation 8.26 is linear with

$$r_1 = \frac{\sigma_f}{\sigma_g} \tag{8.29}$$

$$r_0 = \mu_f - r_1 \mu_g \tag{8.30}$$

This transformation results in equalizing the local mean and standard deviation of the transformed image patch $g'(x,y)$ to μ_f and σ_f respectively.

In this thesis project Wallis filtering was performed during each iteration of the Least Squares Matching process. Figure 8.14 shows an example of dissimilar image patches f and g. Note that $g(x,y)$ has been shifted down in the y-direction relative to $f(x,y)$, resulting in most of the greylevels being in the lower end of the greyscale spectrum as opposed to the greylevels of $f(x,y)$ which are concentrated in the higher end of the greyscale spectrum.

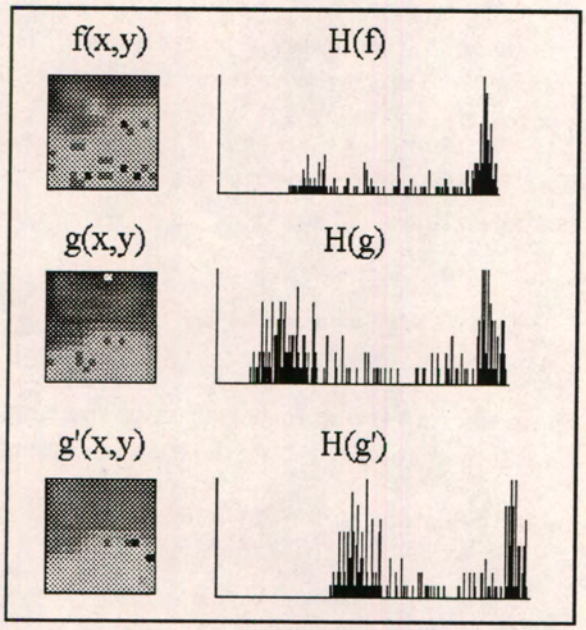


Figure 8.14: Radiometric Corrections for Dissimilar Image Patches

The relatively large differences in the mean values and standard deviation values between f and g result in a large shift factor r_0 of 63.287 greyvalues and a gain factor of 0.766. This results in shifting then compressing the histogram $H(g)$ of g to concentrate it in the higher end of the histogram $H(g')$, shown in figure 8.14.

Table 8.1 shows the radiometric correction parameters for the example of figure 8.14 before the first iteration and after the last iteration of the Least Squares matching process. The mean and standard deviation values of the reference patch $f(x,y)$ are constant, where the mean μ_g of $g(x,y)$ increases during the iterations, approaching the mean μ_f. The standard deviation does not show significant change due to the linear nature of the transformation. Note that due to the matching process the shift parameter r_0 has been reduced by a significant amount from 63 to 35 greylevels.

	First Iteration	Last Iteration
μ_f	146.967	
σ_f	50.319	
μ_g	109.244	138.955
σ_g	65.692	62.240
r_0	63.287	34.625
r_1	0.766	0.808

Table 8.1: Radiometric Correction Parameters for Dissimilar Image Patches

Figure 8.15 shows an example of the radiometric corrections for similar image patches $f(x,y)$ and $g(x,y)$. Similar in this case means that there is little radiometric and geometric distortion between the reference image patch $f(x,y)$ and the candidate image patch $g(x,y)$. The transformed patch $g'(x,y)$ is approximately the same as $g(x,y)$, with a small shift r_0 of 11.286 greylevels and a gain factor r_1 of 0.978. The histograms $H(..)$ of the image patches f, g and g' are also shown. Note that the histogram $H(g)$ does not differ significantly from the histogram $H(g')$.

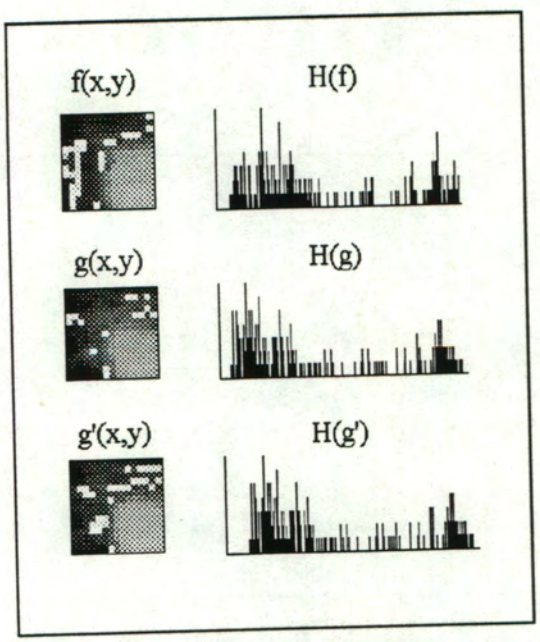


Figure 8.15: Radiometric Corrections for Similar Image Patches

Table 8.2 shows the calculated mean and standard deviation values used in the Wallis filter of the example shown in figure 8.15. These values were calculated before the first iteration of the Least Squares Matching scheme. Note the similarity between the mean values and standard deviation values for f and g.

μ_f	89.291
σ_f	62.450
μ_g	80.569
σ_g	64.503
r_0	11.286
r_1	0.968

Table 8.2: Radiometric Correction Parameters for Similar Image Patches

For this example the Wallis filter has no significant effect on the outcome of the matching process. This is as expected due to the similarity in the image patches.

The reader is referred to figure 8.16 a) for plot of the radiometric parameters r_0 and r_1 as a function of the iteration number n for the example shown in figure 8.14. After each iteration the Wallis filter is applied and the parameters r_0 and r_1 updated. The gain factor r_1 does not change significantly from 0.766 to 0.808 whereas the shift parameter r_0 decreases gradually from 63 to 35 greylevels. Note that only in an ideal case with zero noise and a perfectly linear transformation between f and g will r_0 converge to zero and r_1 converge to one.

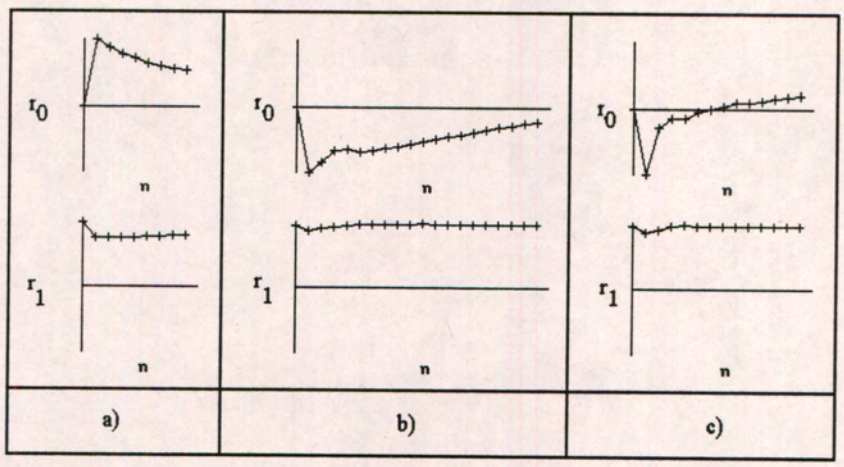


Figure 8.16: Radiometric Corrections Parameters as a Function of Iteration Number n

Figure 8.16 b) and c) show the parameters r_0 and r_1 as a function of the iteration number for the example shown in figure 8.6. After 20 iterations the Least Squares matching scheme hadn't converged sufficiently, resulting in a gradual change of r_0 to zero with r_1 not changing significantly, shown in figure 8.16 b). In figure 8.16 c) the plots of r_0 and r_1 are shown for the same example, this time converging due to using the $L1$-norm as discussed in section 8.1.

8.3 Feature Geometry Constrained LSM

As described in chapter 7, the relative orientation between the left and right images is obtained by using the centre of mass points of the matching features as corresponding points. The next step in the matching process involves matching the known corner points of the matching features. These matched corner positions are then used to update the relative orientation.

8.3.1 Matching Feature Corner Points

Feature corner points are matched using a combination of epipolar line conditions and Least Squares Matching. For each corner of each matched feature in the left image the epipolar line is calculated and the corner position of the matching feature that falls on this line is found (see figure 8.17). This corner position is used as the first approximation to the Least Squares Matching, which refines the matching corner position to the sub-pixel level. One problem that can arise from this approach is when two or more corners fall on the same epipolar line (figure 8.17 b). In this case the correct corner position has to be distinguished from other candidates.

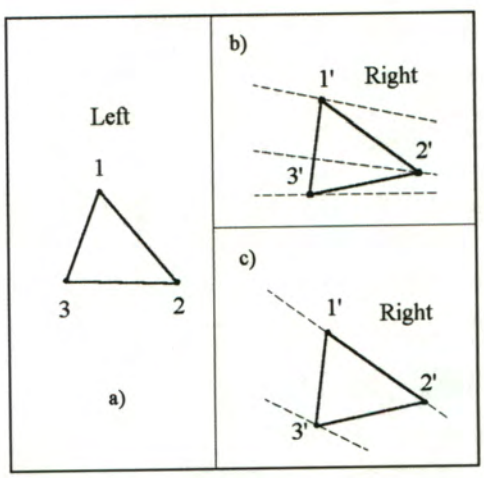


Figure 8.17: a) Reference image feature, b) Unique corner matching candidates and c) Ambiguous corner matching candidates

Two ways of solving this ambiguity were investigated:

Proximity to the Epipolar Line

The corner positions are defined at the pixel-level, and the initial search tries to find the corner position that is "close enough" to the sub-pixel epipolar line. If the epipolar line is represented in the continuous case as

$$y = m\,x + c \tag{8.31}$$

where x and y are continuous variables, then the criteria for checking if a point $P_1(x_1, y_1)$ at discrete location (x_1, y_1) falls on the epipolar line is to check if the shortest distance d between the point P_1 and the epipolar line is less than a preset threshold. This threshold depends on the accuracy of the relative orientation calculation.

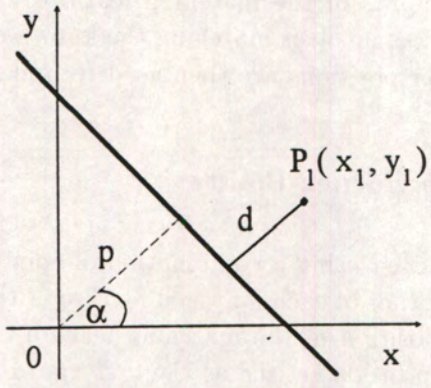


Figure 8.18: The shortest distance between a line and a point

The line equation can be rewritten in the form

$$x\,cos\alpha + y\,sin\alpha = p \tag{8.32}$$

where the angle α and the constant p are as shown in figure 8.18.

The angle α can be calculated using the known slope m of the line as

$$\alpha = \frac{\pi}{2} - tan^{-1}(m) \tag{8.33}$$

assuming the function $tan^{-1}()$ returns the correct angle in the first quadrant. Care should be taken with this function when the epipolar lines become vertical, which should not occur with a good geometric layout between the left and right images.

The constant p can be calculated next by using a known point on the line. In this project either the point (x_1, y_1') or (x_1', y_1) was used, depending on the slope of the line, where

$$y_1' = m\,x_1 + c \tag{8.34}$$

and

$$x_1' = \frac{(y_1 - c)}{m} \tag{8.35}$$

Finally the shortest distance d from the point $P_1(x_1, y_1)$ to the epipolar line can be calculated using the formula

$$d = x_1 cos\alpha + y_1 sin\alpha - p \tag{8.36}$$

The relative orientation obtained from matching the centre of mass points of the matched features can only be seen as a first approximation. As discussed in chapter 7, the centre of mass points do not necessarily exactly match, especially for close range applications.

Using the proximity of the corner points to the epipolar line as a distinguishing measure was considered to be insufficient, as the accuracy of the epipolar line is not high enough.

Area Correlation

The method adopted in this research project to solve ambiguities uses the *normalized 2D cross correlation* function, which calculates the similarity between two image patches.

We can calculate the cross correlation between discrete image $g[n_1, n_2]$ of size M x N and sub-image $h[n1, n2]$ of size J x K with $J \leq M$ and $K \leq N$ (figure 8.19) as

$$\Phi_{gh}[k_1, k_2] = \frac{\sum_{n_1}\sum_{n_2}(g[n_1, n_2] - \bar{g}[n_1, n_2])(h[n_1 - k_1, n_2 - k_2] - \bar{h})}{\sqrt{\sum_{n_1}\sum_{n_2}(g[n_1, n_2] - \bar{g}[n_1, n_2])^2 \sum_{n_1}\sum_{n_2}(h[n_1 - k_1, n_2 - k_2] - \bar{h})^2}} \tag{8.37}$$

where $k_1 = 0, 1..M - 1, k_2 = 0, 1, N - 1$ and the summation is taken over the image region where g and h overlap. The value $\bar{h}$,the average grey level over the sub-image h, is computed once whereas the average grey level $\bar{g}[n_1, n_2]$ is computed for each area of overlap between g and h.

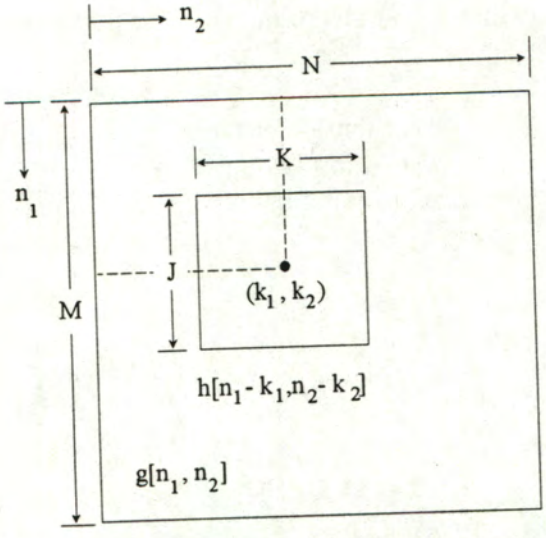


Figure 8.19: Area Correlation of $g[n_1, n_2]$ and $h[n_1, n_2]$ at point (k_1, k_2)

See figure 8.20 for an example of the area correlation function applied to the image g with

the text **"Virtual Reality"**, where the sub-image h is patch containing the letter **"a"**. The output result shows a high correlation between the search patch and the two instances of the letter "a" in the image. Note also that a high correlation was found between the search patch and the letter "e", which does illustrate the shortcomings of using area correlation. The high correlation with the letter "e" can be expected, as the structure of the letters in the image is very similar. The correlation values have been scaled to a greyscale in the range 0..255, thus the points of high correlation are indicated in white (255) and points of poor correlation in darker shades of grey.

If we know that we are looking for only one instance of the letter "a", we would find the correct position by locating the point in the output image that has the highest correlation coefficient.

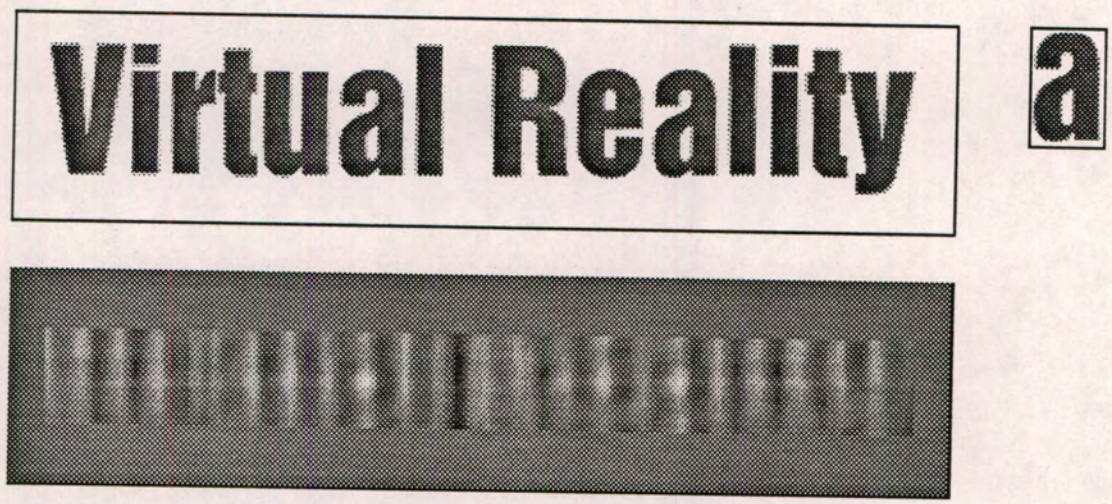


Figure 8.20: Area Correlation of sub-image "a" with the image "Virtual Reality"

Rosenfeld and Kak [85] analyze equation 8.37 using the theory of Least Squares and the *Cauchy-Schwartz* inequality. The analysis is based on observing the squared difference between two non-negative, continuous functions $g(x,y)$ and $h(x,y)$ over an area A. We observe the difference as

$$\iint_A (g-h)^2 = \iint g^2 + \iint h^2 - 2\iint gh \tag{8.38}$$

The autocorrelation functions $\iint g^2$ and $\iint h^2$ are fixed and positive thus the error function $\iint_A (g-h)^2$ is small only if $\iint gh$ is large. This shows that the function $\iint gh$ can be used as a measure of the match between g and h.

The same conclusion can be reached by using the *Cauchy-Schwartz* inequality, that states that for g and h non-negative we always have

$$\iint gh \leq \sqrt{\iint g^2 \iint h^2} \tag{8.39}$$

with equality holding if and only if $h = cg$ for some constant scaling factor c.

The analogous case for a digital system is

$$\sum_{n_1}\sum_{n_2} g[n_1,n_2]h[n_1,n_2] \leq \sqrt{\sum_{n_1}\sum_{n_2} g[n_1,n_2]^2 \sum_{n_1}\sum_{n_2} h[n_1,n_2]^2} \tag{8.40}$$

In section 8.2 on the radiometric corrections between the left and right images we modelled the transformation between the left and right greylevels as a shift and a scale i.e. $h = r_1 g + r_0$ where r_0 and r_1 are constants. Subtracting the mean values $\bar{g}$ and $\bar{h}$ and dividing by the autocorrelations normalizes equation 8.37 such that it still is scaled between -1 and 1 for any scale factor r_1 and translation r_0.

As discussed in section 6.1.2 on feature matching by correlation of the $d\phi(s)$ plots, the correlation function can be obtained from the frequency spectrums $G(\omega_1,\omega_2)$ and $H(\omega_1,\omega_2)$ of g and h.

To match the corner positions , image patches g and h of 15 x 15 pixels each were correlated, with g centered on the left feature corner and h centered on the right feature corner being tested. As the two sub-images are the same size, only one cross-correlation is calculated. The normalized cross-correlation values obtained from the cross-correlation of each of the corner candidates in the right feature are compared and the highest correlation value is accepted.

See figure 8.21 for an example of corner detection using area correlation. For the triangular feature in figure 8.21 a) the corner detection algorithm correctly matched all three corners. Shown below the features is a zoomed view of the 15x15 image patches used to calculate the area correlation between corner 2 of the left image and corner 2 of the right image. The polygon feature on the right in figure 8.21 b) has undergone large geometric distortions, and the corner matching algorithm could only find 3 out of the 7 matching corners. Shown below the features is a zoomed view of the 15x15 image patches used to calculate the area correlation between corner 7 of the left image and corner 0 of the right image.

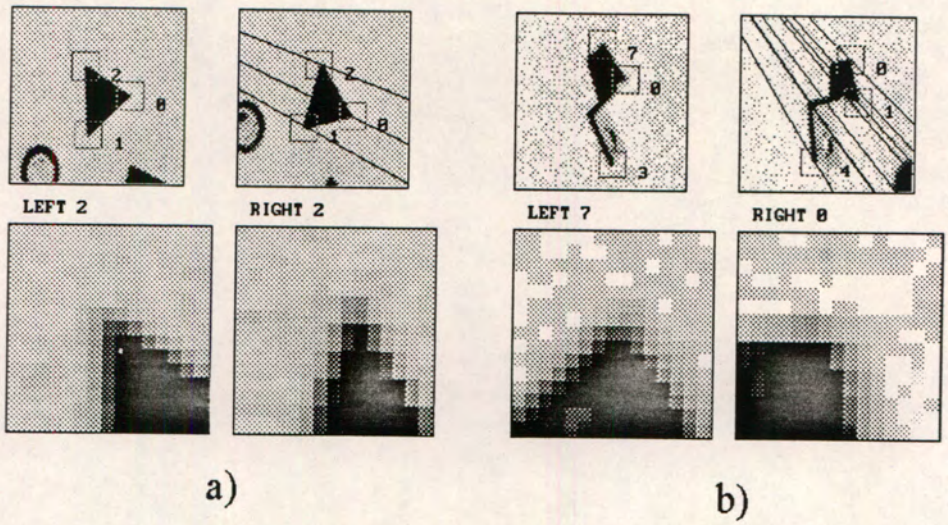


Figure 8.21: Corner detection for two features using area correlation

Being unsuccessful in matching all the corners is not due to the area correlation function, but due to the fact that in the image used the epipolar line was not of sufficient accuracy to correctly find the correct corner candidate to be correlated. The reason for this is twofold. Firstly, it was subsequently discovered that the test image used had bad image geometry, making the relative orientation calculation very sensitive to small errors in the corresponding image coordinates. Secondly, the relative orientation in the region of this polygon feature is erroneous. The centre-of-mass points of matching features were assumed to be corresponding points that were used to calculate the relative orientation. For this feature the corresponding point is not exactly correct as the feature has undergone large geometric distortions . These distortions are due to the feature being high relative to the viewing height, which exposes the weakness of the planar approximation in a case when the object heights are not small relative to the viewing height. As stated earlier in the section on feature matching, the planar approximation does hold for aerial images and close-range images in which the object heights are small compared to the viewing height. The corresponding centre-of-mass coordinates for this feature causes an outlier, which does effect calculations on points close to this outlier.

Solving ambiguities using Least Squares Matching as described in section 8.1 was also investigated. The LSM operation allows rotation and scaling of the image patch, which could in this case result in a successful match for either of the candidates . Examining the residuals of the matches does not necessarily distinguish between the correct and the incorrect matches. If the rotation between the left and the right image is large however, the LSM approach could be better.

If the LSM approach is used, some constraints must be placed on the rotation parameters of the affine model. The constraints can be enforced by either weighting these parameters heavily in the A^TPA matrix , which will ensure that the parameters change slowly , or by actually checking the rotation parameters at each step and stop iterating if the rotations become too big. Here, as is often the case in having to quantify values, it is difficult to decide *a priori* exactly how much the rotation parameters must be weighted or how much rotation

can be tolerated?

For aerial photography and close-range applications without a large rotation between them the area correlation function gave better results in correctly finding the matching corner points. Once these corner points were found the result was used to refine the corner matching points to sub-pixel level using LSM. The results of the LSM gave an extra set of sub-pixel matching points to the already matched feature centre of mass points. These two data sets are now combined to update the relative orientation, which will give a more accurate measure of the relative orientation parameters.

Refining Corner Coordinates

In this thesis project the corner coordinates in the reference image were refined using the intersection between two best-fitting straight lines. Heuristic knowledge about corners was used by assuming that corners such as the corner of a building or a manufactured part is formed by the intersection of two straight lines in the vicinity of the corner. This conforms to a piecewise-linear approximation to the feature outlines.

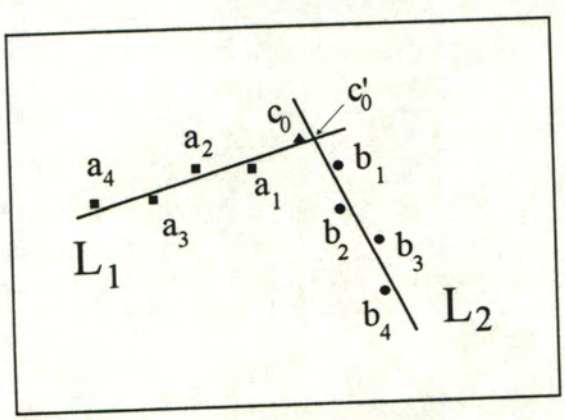


Figure 8.22: Corners from line intersections

In figure 8.22 the new corner position C_0' is defined as the intersection between the two best-fitting straight lines L_1 and L_2. The line L_1 is fit between the sub-pixel edge points $a_1 ... a_4$ and the original corner point c_0, and the line L_2 is fit between the sub-pixel edge points $b_1 ... b_4$ and the original corner point c_0.

Note that in this example only four points on either side of the corner position c_0 were used to find the new corner position. A corner point can be found at the end of a curved line and fitting a line using too many edge points will have a negative effect on the piecewise-linear approximation of a curve.

The "integrity" of a best-fitting straight line can be checked by observing the term σ_0 of the line fit. If this term is larger than a preset threshold value then the data points poorly represent a straight line and the new corner point is rejected. If both lines L_1 and L_2 have σ_0 less than this threshold value then the linear regressions are accepted as "good enough" and the new corner position C_0' is accepted.

As the corner position C_0 or C_0' is at a sub-pixel level, the reference image patch $f(x,y)$ can initially be resampled to be centered at this position. Experimentation showed that as the candidate image patch $g(x,y)$ is resampled at each iteration of the LSM process, resampling $f(x,y)$ had no significant effect on the outcome of the matching process.

See the appendix for the Least-Squares solution to the linear regression problem.

8.4 Object Coordinate Calculation

The final step in the matching scheme is the calculation of the 3-D object coordinates for each corresponding image point. The object coordinates are calculated using the collinearity equations introduced in section 2.2.

The collinearity equations are repeated in equations 8.41 and 8.42

$$u = f\frac{r_{11}(X - X_0) + r_{12}(Y - Y_0) + r_{13}(Z - Z_0)}{r_{31}(X - X_0) + r_{32}(Y - Y_0) + r_{33}(Z - Z_0)} \tag{8.41}$$

$$v = f\frac{r_{21}(X - X_0) + r_{22}(Y - Y_0) + r_{23}(Z - Z_0)}{r_{31}(X - X_0) + r_{32}(Y - Y_0) + r_{33}(Z - Z_0)} \tag{8.42}$$

where (u,v) are the image coordinates reduced to the principal point and $(X - X_0, Y - Y_0, Z - Z_0)$ are the 3-D object coordinates (X,Y,Z) reduced to the perspective centre. The camera constant f is known and so are the coefficients r of the rotation matrix $R(\omega,\phi,\kappa)$ where

$$R(\omega,\phi,\kappa) = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{8.43}$$

If the absolute orientation is known, the three object coordinates can be directly calculated using 3 of the 4 collinearity equations obtained from a corresponding image coordinate pair. Control points are used to reference the image coordinate systems to global coordinate system. In this thesis project only the relative orientation is known, and the object coordinates can only be calculated in the coordinate system defined during the relative orientation discussed in chapter 7. The perspective centre of the left camera was taken as the origin, and the base Bx between the cameras in the x-direction was set equal to one. The rotation matrix for the left image reduces to the identity matrix I because the rotations are all zero at the origin. The identity matrix I is defined as

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{8.44}$$

For the left image the collinearity equations 8.41 and 8.42 simplify to

$$u_1 = f_1 \frac{X}{Z} \tag{8.45}$$

$$v_1 = f_1 \frac{Y}{Z} \tag{8.46}$$

If the relative orientation yielded a solution where the perspective centre of the right image is (X_2, Y_2, Z_2) and the 3 rotation angles about the x, y and z axes are ω, ϕ and κ respectively, the rotation matrix for the right image is given by $R(\omega, \phi, \kappa)$ as represented in equation 8.43.

After simplification of the collinearity equations for the right image, the equations reduce to

$$A_1X + A_2Y + A_3Z = A_1X_2 + A_2Y_2 + A_3Z_2 \tag{8.47}$$

$$B_1X + B_2Y + B_3Z = B_1X_2 + B_2Y_2 + B_3Z_2 \tag{8.48}$$

where

$$\begin{aligned} A_1 &= u_2r_{31} - f_2r_{11} \\ A_2 &= u_2r_{32} - f_2r_{12} \\ A_3 &= u_2r_{33} - f_2r_{13} \\ B_1 &= v_2r_{31} - f_2r_{21} \\ B_2 &= v_2r_{32} - f_2r_{22} \\ B_3 &= v_2r_{33} - f_2r_{23} \end{aligned}$$

By adding one of the collinearity equations for the left image to this system of equations we can solve for the (X, Y, Z) object coordinates, which will involve inverting a 3x3 matrix. To reduce the size of the matrix to be inverted to solve for these equations to a 2x2 matrix, substitute

$$Z = f_1 \frac{X}{u_1} \tag{8.49}$$

into equations 8.47 and 8.48, which then reduces to

$$\begin{pmatrix} A_1' & A_2 \\ B_1' & B_2 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \tag{8.50}$$

where

$$
\begin{aligned}
A_1' &= A_1 + A_3\frac{f_1}{u_1} \\
B_1' &= B_1 + B_3\frac{f_1}{u_1} \\
C_1 &= A_1X_2 + A_2Y_2 + A_3Z_2 \\
C_2 &= B_1X_2 + B_2Y_2 + B_3Z_2
\end{aligned}
$$

From this reduced equation 8.50 the X and Y coordinates can be calculated as

$$
\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} A_1' & A_2 \\ B_1' & B_2 \end{pmatrix}^{-1} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \tag{8.51}
$$

and substituted back into equation 8.49 to find the Z component.

8.4.1 Refining Object Coordinates

The accuracy of the object coordinate calculations depends on the accuracy of the corresponding image points and the point field geometry. The object coordinates resulting from a corresponding pair of image points can be refined using a Least Squares model.

From the collinearity equations 8.41 and 8.42 let

$$
\begin{aligned}
F_1^u &= u_1 \\
F_1^v &= v_1 \\
F_2^u &= u_2 \\
F_2^v &= v_2
\end{aligned}
$$

The collinearity equations for the left image can be linearized using the first derivative terms of the Taylor series:

$$
u_1 + V_{u_1} = F_1^{u(0)} + \frac{\partial F_1^u}{\partial X}dX + \frac{\partial F_1^u}{\partial Y}dY + \frac{\partial F_1^u}{\partial Z}dZ \tag{8.52}
$$

$$
v_1 + V_{v_1} = F_1^{v(0)} + \frac{\partial F_1^v}{\partial X}dX + \frac{\partial F_1^v}{\partial Y}dY + \frac{\partial F_1^v}{\partial Z}dZ \tag{8.53}
$$

where V_{u_1} and V_{v_1} are the corrections to the image coordinates.

Equations 8.52 and 8.53 can be rearranged as

$$V_{u_1} = \frac{\partial F_1^u}{\partial X} dX + \frac{\partial F_1^u}{\partial Y} dY + \frac{\partial F_1^u}{\partial Z} dZ - (u_1 - F_1^{u(0)}) \tag{8.54}$$

$$V_{v_1} = \frac{\partial F_1^v}{\partial X} dX + \frac{\partial F_1^v}{\partial Y} dY + \frac{\partial F_1^v}{\partial Z} dZ - (v_1 - F_1^{v(0)}) \tag{8.55}$$

(8.56)

These are observation equations with observations (u_1, v_1) and unknowns dX, dY and dZ. The constants $F_1^{u(0)}$ and $F_1^{v(0)}$ are the values of F_1^u and F_1^v calculated at the first approximation of the 3-D object coordinate. In a similar fashion the observation equations for the right image can be computed, resulting in a linear set of equations in the form

$$\vec{v} = A\vec{X} - \vec{l} \tag{8.57}$$

where $\vec{v}$ is the vector of corrections to the image coordinates for an object point imaged as an image pair.

$$\vec{v} = \begin{pmatrix} V_{u_1} \\ V_{v_1} \\ V_{u_2} \\ V_{v_2} \end{pmatrix} \tag{8.58}$$

The design matrix A consists of the partial derivatives of the collinearity equations with respect to the object coordinates

$$A = \begin{pmatrix} \frac{\partial F_1^u}{\partial X} & \frac{\partial F_1^u}{\partial Y} & \frac{\partial F_1^u}{\partial Z} \\ \frac{\partial F_1^v}{\partial X} & \frac{\partial F_1^v}{\partial Y} & \frac{\partial F_1^v}{\partial Z} \\ \frac{\partial F_2^u}{\partial X} & \frac{\partial F_2^u}{\partial Y} & \frac{\partial F_2^u}{\partial Z} \\ \frac{\partial F_2^v}{\partial X} & \frac{\partial F_2^v}{\partial Y} & \frac{\partial F_2^v}{\partial Z} \end{pmatrix} \tag{8.59}$$

The vector of unknowns $\vec{X}$ in equation 8.57 is the corrections to the object coordinates

$$\vec{X} = \begin{pmatrix} dX \\ dY \\ dZ \end{pmatrix} \tag{8.60}$$

and the l-terms depend on the image coordinate observations and the collinearity equations at the initial approximation of the object coordinates

$$\vec{l} = \begin{pmatrix} u_1 - F_1^{u(0)} \\ v_1 - F_1^{v(0)} \\ u_2 - F_2^{u(0)} \\ v_2 - F_2^{v(0)} \end{pmatrix} \tag{8.61}$$

The corrections to the object coordinates are found using the Least Squares solution which finds $\vec{X}$ as

$$\vec{X} = (A^T P A)^{-1}(A^T P l) \tag{8.62}$$

where P is the weight matrix used to assign weights to the observations. In this research project equal weights were assigned to each observation, resulting in a weight matrix equal to the identity matrix I.

The XYZ object coordinates are updated with the corrections and the process is iterated until the convergence criteria has been met. In this thesis project the iterations were stopped when the vector-sum of the corrections became less than a preset threshold value. After iteration n the new object coordinates are calculated as

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}^{n+1} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}^{n} + \begin{pmatrix} dX \\ dY \\ dZ \end{pmatrix}^{n} \tag{8.63}$$

The new values (X, Y, Z) for the object coordinates are used to calculate the derivatives in the A matrix and the $F^{u(0)}$ and $F^{v(0)}$ constants in the residual vector $\vec{l}$.

If one rewrites the collinearity equations 8.41 and 8.42 as

$$F^u = f\frac{M_1}{M_3} \tag{8.64}$$

$$F^v = f\frac{M_2}{M_3} \tag{8.65}$$

then the partial derivatives of equations 8.64 and 8.65 can be calculated as follows:

$$\frac{\partial F^u}{\partial X} = f\frac{(r_{11}M_3 - M_1 r_{31})}{M_3^2} \tag{8.66}$$

$$\frac{\partial F^u}{\partial Y} = f\frac{(r_{12}M_3 - M_1 r_{32})}{M_3^2} \tag{8.67}$$